HELP US OUT: READER SURVEY INSIDE...PAGE 81

DOS RESOURCE GUIDE

The PC Productivity Magazine

AN IDG COMMUNICATIONS PUBLICATION
NUMBER 5
$4.95
CANADA $5.95
UK £3.00

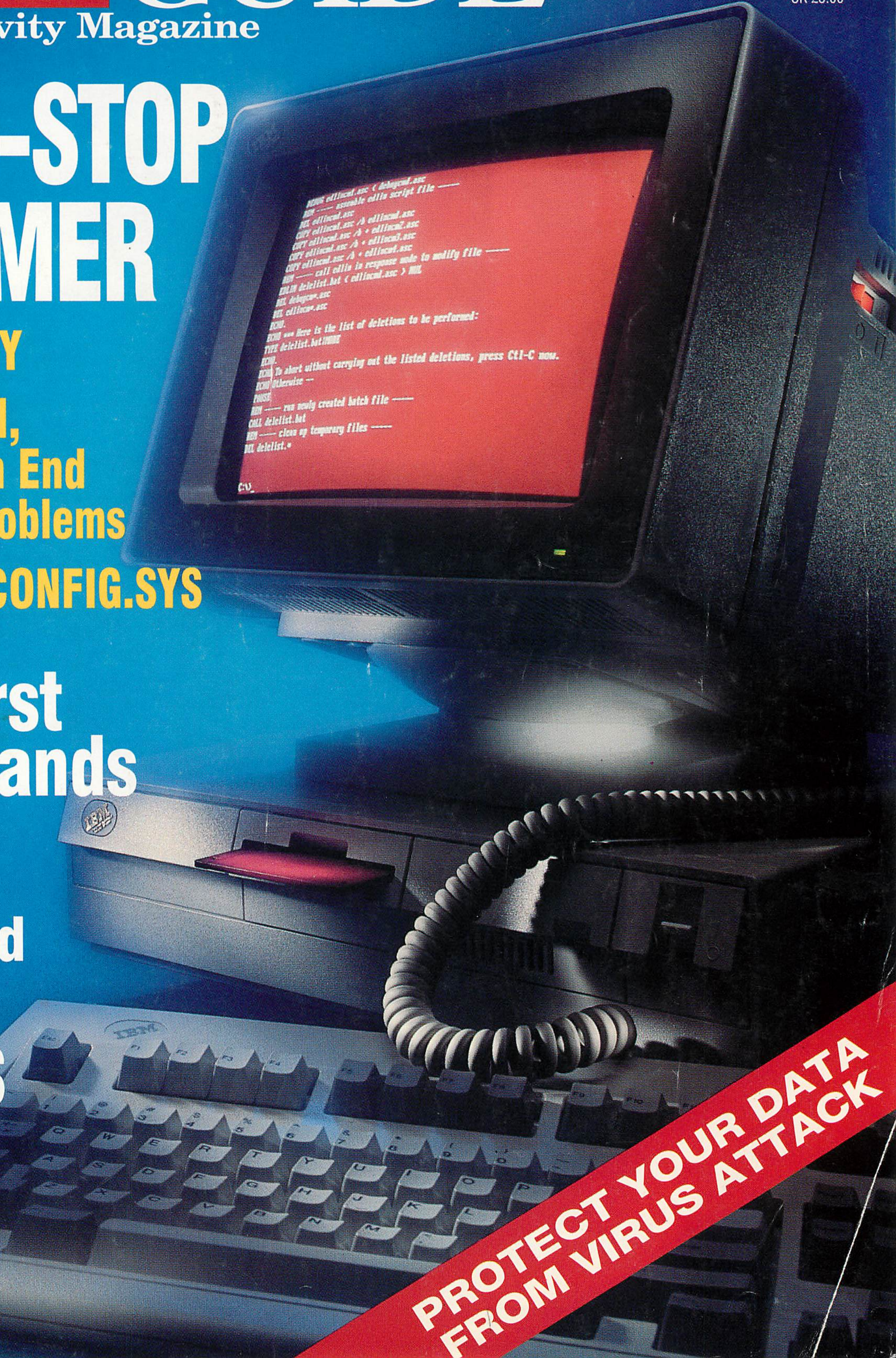
THE ONE-STOP DOS PRIMER

- All About DOSKEY
- APPEND, ASSIGN, And SETVER Can End Compatibility Problems
- Fine-tune Your CONFIG.SYS

The 10 Worst DOS Commands

DEBUG
Almost Defined

INSIDE:
15 DOS TIPS

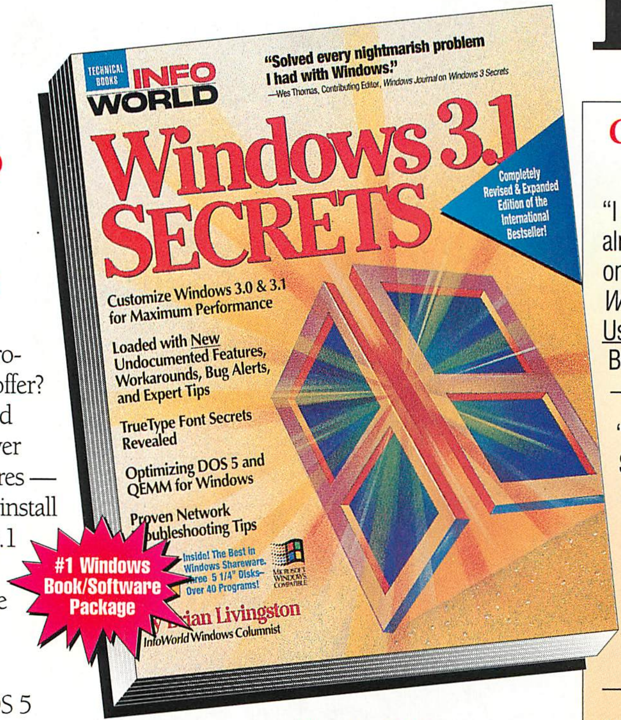


PROTECT YOUR DATA FROM VIRUS ATTACK

AN IDG COMMUNICATIONS PUBLICATION
98
0 74470 78613 9

Completely Revised and Expanded!

Secret Weapons



TECHNICAL BOOKS
INFO WORLD
"Solved every nightmarish problem I had with Windows."
—Wes Thomas, Contributing Editor, Windows Journal on Windows 3 Secrets
Windows 3.1 SECRETS
Completely Revised & Expanded Edition of the International Bestseller!
Customize Windows 3.0 & 3.1 for Maximum Performance
Loaded with New Undocumented Features, Workarounds, Bug Alerts, and Expert Tips
TrueType Font Secrets Revealed
Optimizing DOS 5 and QEMM for Windows
Proven Network Troubleshooting Tips
Inside! The Best in Windows Shareware. Three 5 1/4" Disks—Over 40 Programs!
#1 Windows Book/Software Package
Brian Livingston
InfoWorld Windows Columnist

Let *InfoWorld's* Windows Expert Help Put You On The Winning Side with Windows 3.1 Secrets

Are you in a constant battle with Windows instead of reaping the productivity rewards it's supposed to offer? *InfoWorld's* Windows columnist and guru Brian Livingston gives you over 980 pages of insider tips and features — secret weapons that will help you install and configure Windows 3.0 and 3.1 for maximum performance. And over 40 programs representing The Best in Windows Shareware.

Windows 3.1 Secrets Gives You:

- Expert advice on optimizing DOS 5 and QEMM with Windows 3.0 & 3.1
- A complete guide to converting your company to a Windows network
- Secrets of TrueType™ Fonts including Multiple Masters
- Valuable tips on all the new features in Windows 3.1

$39.95 US/ $52.95 Canada, 980 pages, three 5 1/4" Disks, ISBN 1-878058-43-6

Critical Acclaim for the Original Edition, *Windows 3 Secrets:*

"I don't care how many Windows books you already have; get this one. If you have this one, it's not likely you'll need another. *Windows 3 Secrets* hands down wins the User's Choice Award for the Best Windows Book of 1991."
— Jerry Pournelle, *BYTE Magazine*, April 1992

"There is just too much great stuff here... Simply stated, this book is a must for serious programmers."
— Mitchell Waite, *BasicPro Magazine*

"*Windows 3 Secrets* is a storehouse of facts and figures that will help anyone to get more out of Windows...Get Brian Livingston's book."
— Tom Swan, *PC Techniques*

"*Windows 3 Secrets* includes more undocumented features, tips, and techniques than any other Windows book."
— David Angell & Brent Heslop, *Computer Currents*

Also Available: *PC World DOS 5 Complete Handbook*

This book has been called "the definitive guide to DOS." It gives you a clear, detailed view of DOS 5 — the Shell program, running programs and batch files effectively, protecting data, advanced file operations, customization — and includes FREE Norton Commander® Special Edition software, which makes DOS more intuitive with a graphical user interface.

$29.95 US/$39.95 Canada, 616 pages, one 5 1/4" disk, ISBN: 1-878058-13-4

The Best in Windows Shareware

MICROSOFT® WINDOWS™ COMPATIBLE

Three 5 1/4" disks packed with over 40 shareware programs — over 6 megabytes of software! Including file and program management, graphics, text editing, communications, and utilities!

VIRUS PROTECTION
Viruscan & Cleanup

FILE AND PROGRAM MANAGEMENT
File Commander, Launch, RecRun, RunProg, Task Manager & Desktop Navigator

GRAPHICS
Icon Manager, MetaPlay, Paint Shop, WinGIF

COMMUNICATIONS
ComReset, Unicom

MAJOR APPLICATIONS
WindBase

TEXT EDITING AND SEARCHING
Hunter, WinEdit, WinPost

GAMES
Chess, Klotz, Lander, Puzzle

UTILITIES
Big Cursor, BizWiz, ClockMan, EDOS, SnagIt, Superload, Trash Can, Whiskers, WinBatch, WinExit, Windows Unarchive, WordBasic Macros, W. BAT, Visual Basic Programs, and more!

IDG BOOKS

The World of Computer Knowledge.

IDG Books Worldwide
155 Bovet Road, Suite 610
San Mateo, CA 94402
(415) 312-0650 (800) 762-2974

Choose Your Secret Weapon!

Available in your local bookstore's computer book section.

Or order now by calling
1-800-762-2974
7 a.m. to 5 p.m. PST or complete and mail the form below:

Yes! Please send me:	Qty	Price	Tax*	Ship	Total
Windows 3.1 Secrets	____	$39.95	____	$4.00	____
PC World DOS 5 Complete Handbook	____	$29.95	____	$4.00	____

UPS Shipping & Handling ($4 per book*) CA Residents add applicable sales tax. IN residents add 5% tax. Canadian residents use Canadian prices and add 7% GST tax.

Name ______________________
Company ______________________
Address ______________________
City, State, Zip ______________________
Phone ______________________

Payment method (payment with credit card number or check must accompany your order):
☐ Check enclosed ☐ Visa/MasterCard/American Express
Card # ______________________
Expiration Date ______________________
Signature ______________________

Inquire about rush delivery service (an extra charge will apply). Corporate Trainers quantity discounts available.

DRG

DOS RESOURCE GUIDE

EDITOR
Michael J. Comendul

MANAGING EDITOR
Marilyn McMaster

ARTICLE EDITORS
Patricia Cummings
Beth Jala

COPY EDITOR
Peg LePage

CONTRIBUTING EDITOR
John Wolfskill

CONTRIBUTING WRITERS
Hardin Brothers, Neil Burnett, Dan Gookin, Brett Glass, James Langdell, Steve Nebel, Kay Yarborough Nelson, Jack Nimersheim, Alan L. Zeichick

DESIGN & PRODUCTION
Lou Ann Morin

ELECTRONIC PUBLISHING
Lindy P. Greenwood

Back issues of *DOS Resource Guide* are available for $5.50, postage paid. Call: 1-800-238-6580 (9 a.m. to 5 p.m. EST).

If you are interested in advertising your products or services in *DOS Resource Guide*, call Jeff DeTray, publisher, (603) 924-0249.

How to contact us: *DOS Resource Guide* welcomes your comments, DOS tips, and article submissions. Submit them care of our offices, or by CompuServe: 75300,2361; or MCI Mail: 469-7260.

DOS Resource Guide (ISSN: 1056-7364) is published quarterly by Business Computer Publishing Inc., an IDG company. Business offices at 80 Elm St., Peterborough, NH 03458. (603) 924-0100. Entire contents copyright 1992, by Business Computer Publishing Inc. No part of this publication may be printed or otherwise reproduced without written permission from the publisher. *DOS Resource Guide* assumes no responsibility for damages due to errors or omissions in our articles.

Business Computer Publishing, Inc.

PRESIDENT/PUBLISHER
Jeff DeTray

EDITORIAL DIRECTOR
Michael J. Comendul

IDG COMMUNICATIONS *DOS Resource Guide* is a publication of International Data Group, the world's leader in information services on information technology.

DRG EDITORIAL

Understanding DOS And the People Who Use It

How do you learn about DOS? As the staff of *DOS Resource Guide* ambles along in an attempt to better serve its readers, we make changes. In this issue, for example, we add a box in our DOS articles that summarizes the command under discussion, tells you in what version it was introduced and last updated, and says whether it is internal or external to DOS.

The difference between external and internal commands is not small, although it is becoming less important. Internal DOS commands will run from the DOS prompt no matter how you get there. Some of the internal DOS commands are COPY; DATE (a command that started life as an external command); and MD, CD, and RD for making, changing, and removing directories.

Much of DOS's power, however, lies in its external commands, such as XCOPY, FIND, and FORMAT. Calling these commands usually requires you to be in the DOS subdirectory. Today, most of us own hard disks set up with DOS's directory in a PATH statement. (PATH's another internal command.) A roadmap to your subdirectories, PATH allows DOS to find an external command from anywhere in your subdirectory structure.

How Microsoft chose certain commands to be internal and others to be external to DOS is a mystery. At least it is to me. You'd think that FORMAT, for example, would be an internal command, but it isn't.

This lack of clarity surrounding DOS leads to general confusion about using the operating system. This confusion fuels publications that purport to help the PC user, but so many software manuals and computer guidebooks are cobbled together from ambiguous phrases and jargon that they have become freakish grammatical artifacts of this "information age."

DOS Resource Guide is trying to make clear what is now the sinuous (often tortured) prose of technology. The magazine is committed to teaching by example the way to solve problems with DOS.

And we're interested in how you learn about DOS. Our reader survey, located at the back of this issue, inquires about your computing habits: your level of comfort with DOS; the hardware and software you own or plan to buy in the coming year; and most important, what kind of articles you need to see in *DOS Resource Guide.*

We'll be listening—just as we were last issue when you asked about subscriptions. With this issue, *DOS Resource Guide* is offering a subscription that will save you some money and, with cooperation from the U.S. Post Office, will guarantee you'll see our next issue.

We've made some other swift business decisions, as well. For instance, we corrected a toll-free number for your orders (and I'll add my apologies to the proprietor of that flower shop in New Jersey who heard from a great many *DOS Resource Guide* readers trying to order back issues instead of Easter bouquets).

We're looking for more opinions on this issue. Let us hear from you.

Michael J. Comendul

DOS RESOURCE GUIDE

DOS

Develop Your PC's Personality 11
CONFIG.SYS and AUTOEXEC.BAT build character—your PC's character, that is.
by Hardin Brothers

Positive Attributes 15
If you make good use of ATTRIB, your files will be safer and backups will go more smoothly.
by James Langdell

A Farewell to Incompatibility 19
Four DOS commands that bring peace to warring hardware and software.
by John Wolfskill

10 Worst DOS Commands 23
One computer journalist's expose on the dogs of DOS.
by Dan Gookin

DOSKEY Revealed 28
Tips and traps on recalling, editing, and saving commands in DOS 5.0.
by Kay Yarborough Nelson

Introducing Debug 34
Part I: Experimentation breeds understanding of this most mysterious of DOS programs.
by Dan Gookin

FYI

Upgrade Now! 40
Every PC compatible should be outfitted with DOS 5.0. Here's why.
by Jack Nimersheim

Understanding Serial Ports 44
What you don't know about serial ports can cause lockups and even damage your computer.
by Brett Glass

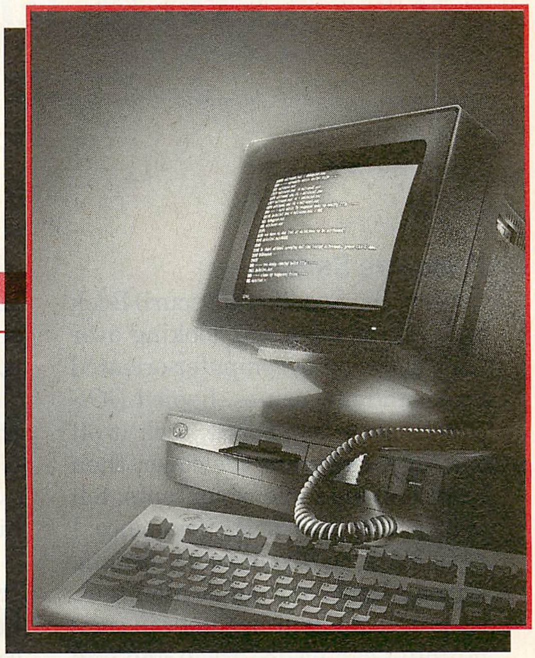
Protect Your Data from Computer Viruses 50
It's not all hype. Disaster could strike your programs and disks if you don't watch out.
by Alan L. Zeichick

Programming from A to Z 56
Part I: Knowing how to program is a powerful skill, and QBasic is an ideal language to start with.
by John Wolfskill

BAT

Box It Up 63
Spruce up reports, program menus...you name it with these simple batch files.
by Neil Burnett

Hard Disk Housekeeping 65
This pair of batch files sweeps away unwanted files—with no muss and no fuss.
by Stephen Nebel



Number 5
1992

DRG

Editorial 1
Letters to the Editor 4
Tips from Readers 5
Best of the Batch 9

TIP

Doing What DOS Doesn't
Creating your own DOS commands with DOSKEY. 12

More Work From One
FOR makes quick work of copying and combining files. 13

Info at Your Fingertips
HELP is at hand in DOS 5.0. 14

A Timely PROMPT
Let DOS help you keep track of time. 22

Memory Magic
How to increase or decrease the size of the DOSKEY buffer. 24

A Directory by Any Other Name
Using SUBST to shorten your PATH statement. 35

Conserving Memory
Remove stack interrupts and save 2K of RAM. 41

Analyzing Your Archives
Display the list of files archived with BACKUP. 42

MORE is Less
Viewing ASCII files one screen at a time. 43

Fooling FORMAT
Use batch files to correctly format your disks—first time, every time. 45

Faster Task Switching
Set up a RAM disk to speed up task switching within the DOS shell. 47

One-Step Directory
Erase files and directories in one step. 51

Safety First
Prevent installation programs from modifying CONFIG.SYS and AUTOEXEC.BAT. 55

Keeping Current
Refreshing the DOS shell's directory information. 66

Quick Command
DOSKEY's function-key shortcuts. 67

Q&A

Starting Up DOS 70
by Hardin Brothers

Files and Directories 71
by James Langdell

Memory Management 72
by Brett Glass

Batch File Programming 73
by John Wolfskill

Hardware 74
by John Wolfskill

DRG LETTERS TO THE EDITOR

It Makes Sense

I was in the Sedalia (Missouri) Book Store three days ago, looking over their selection of computer-oriented books and magazines. I spied *DOS Resource Guide (DRG)*, took it off the rack, looked at the cover, then looked at the table of contents, fell in love, and paid for it. Ever since, I have felt good about *finally* coming across a computer publication that, as a rank amateur/beginner in computing, I can easily grasp and and use. I'm ecstatic. I haven't read one page that isn't informative.

I've had a computer for about a year, but, with work and family, haven't had much time to work at learning computing. So, most of the little I've learned has been by word of mouth and through other computer publications. Yours has been a godsend: easy to read, understand, and absorb. Thanks. I've finally found a publication that makes sense (to me).

John W. Spencer
Knob Noster, Mo.

I have been eagerly awaiting the next issue of *DOS Resource Guide*. My system consists of a Tandy 1000A, Tandy CM-4 monitor, Tandy DMP-132A printer, KALOK 330 hardcard from Perry Computers, Tandy 1200 Plus modem, and 640K of RAM. The BIOS is a Phoenix/Tandy Version 1.01.

As you can see, my system is ancient by any standards. Tandy's latest catalog does not contain any 8088 computers.

I recently upgraded to DOS 5.0, but most of my software is only first or second generation because I can't afford to replace it. I am getting really perturbed, though, that the paint and publishing programs are going more and more to the higher resolutions and offer no downward compatibility.

My point in telling you all this is that I think your *DOS Resource Guide* is needed by people like me even more than those with 486 high-end machines.

Please make it available by subscription. I don't always get to the newsstand when new magazines come out and would hate to miss an issue.

Also, please don't forget the little guy who, for economic or other reasons, may still be in the dark ages of computing.

Ronald E. Sherard Jr.
Plainville, Kan.

Omissions and Corrections

I recently saw issue Number 3 of your magazine in a bookstore and bought it on the chance it might be useful in our Senior Citizens' Computer Club. Upon studying it, I found it a goldmine! Several members rushed out to find copies for themselves.

I did find the usual typo problems in program listings, but they were easily corrected. Program Listing 4 in "Make Your Batch Files Stop, Look, and Listen" (page 68) has spacing problems. The line that reads:

```
:KEY 78
```

should read:

```
:KEY78
```

and the next line should read:

```
:KEY110
```

Earlier on there's another problem. If GETKEY is not in the same directory as the batch file using it, DIRSORT will go directly to sorting. I suggest inserting a CD command immediately prior to the GETKEY command.

The article on ANSI.SYS is great. It will be a permanent reference. I always wondered how to get a white-on-blue screen and how to blank the screen. Now I know!

K. E. Hamilton
Laverne, Calif.

DOS RESOURCE GUIDE welcomes feedback from its readers. Address your DOS questions or comments to:

DOS Resource Guide
80 Elm St.
Peterborough, NH 03458
603-924-0100
603-924-9384 (fax)

You also may contact the editors electronically on MCI (469-7260) or CompuServe (75300-2361).

List and Pause Your Files

Almost all computer magazines provide the equivalent of the FOR loop you presented in the tip " Go For It" (*DRG* #3, page 28) :

```
FOR %A in (*.BAT) DO TYPE %A
```

You mentioned that it works with short files. But, unless your files are extremely short, you will never be able to read them before they scroll off the screen.

I have a very simple batch file, called LIST.BAT, that can serve many purposes. It will type one or more ASCII files, stopping after each page (24 lines) and after each file, so you have time to peek at every line of your files.

You can use it in two ways: as a simple command that you type at the DOS prompt or as a program you invoke from within another batch file. It contains the following six lines:

```
@ECHO OFF
: LIST.BAT
CLS
TYPE %1 | MORE
ECHO.
PAUSE
```

The beauty of using LIST.BAT from the DOS prompt is that you can peruse files without having to type a complex command. Instead of typing a sequence such as:

```
TYPE MYFILE.TXT | MORE
```

you can simply type the following:

```
LIST MYFILE.TXT
```

You also can use LIST.BAT with the FOR command to display a series of files, one after the other. The program ensures that you can view your files 24 lines at a time and pause after each file.

To do this, use a FOR command such as the following:

```
FOR %A IN (*.TXT) DO CALL LIST %A
```

Adding this line to a batch file such as TIPE.BAT:

```
@ECHO OFF
: TIPE.BAT
:LOOP
IF %1!==! GOTO END
FOR %%A IN (%1) DO CALL LIST %%A
SHIFT
GOTO LOOP
:END
```

lets you display files with different filenames and extensions. When using this batch file, you include the names of the files you want to display when you invoke TIPE.BAT. To type a file called MYFILE.DOC, one called NEWFILE.DOC, and any files with a .TXT extension, for example, you would enter this command:

```
TIPE MYFILE.DOC NEWFILE.DOC *.TXT
```

You even can type files from different directories and from different drives. You must specify the drive and directory of the file or files you want to type.

One more point: In the article "COPY: New Tricks for an Old Command" (*DRG* #3, page 30), Lewis Rosenfelder offers a different way to duplicate a disk using the COPY command. But if the source disk contains subdirectories, the XCOPY command is better suited for this task. If you use XCOPY and add an /S switch, the directories of the source disk will be created on the target disk and all files from the source disk will be copied into the corresponding directories of the target disk.

You also can add a /V switch to XCOPY so it will verify the integrity of each file as it is copied to the target disk.

This command also provides a way to copy the contents of one disk to another disk that has a different format. The command to do this is as follows:

```
XCOPY A: B: /S /V
```

This example assumes that you want to copy a file from a disk in drive A to a second disk in drive B.

Claude Dionne
Amos, Quebec

Piggybacking Tips

Being a neophyte in the computer game, albeit a 74-year-old neophyte, I get deeply involved in trying to make the computer do what I want it to do, and I even get successful once in a while. To help myself along the way, I always scan the book racks and the magazine stands for publications that can assist me.

Doing my "scanning," I noticed the premiere issue of your *DOS Resource Guide*, picked it up, had a look through it, and soon came to the conclusion that this new magazine could be really interesting. Now, having read the first issue, I will be looking forward to subsequent editions.

There were two tips that caught my attention very quickly: "Colorful Startups" (page 7) and "Super Prompt" (page 11) . The PROMPT sequences did nothing for me and I use Debug only when I have to, so I combined the two tips by adding to a PROMPT set up as follows:

```
PROMPT $E[s$E[1;53H$T$H$H$H$H$H$H $D$E[u$P$G$E[1;37;44m
```

Adding $E[1;37;44m gives me a white foreground and a blue background, and should I want a different color combination, I can change just the last two numbers. This PROMPT command is in my AUTOEXEC.BAT file.

Just thought I would toss that one in, and again, thanks for a great new magazine.

William G. Scherk
Wescosville, Penn.

Pushing and Popping Directories

I have written a group of batch files that adapt the technique in the tip "Home Please" (*DRG* #1, page 75). Together, they let you create a "stack" of previously used directories as you move around your hard drive, then quickly and easily return to those directories (see Listings 1–5). This group of files is particularly useful if you have numerous structures and jump around a lot.

The first file, PUSH.BAT, saves up to eight directories. When you're in a directory to which you may want to return later, type:

```
PUSH
```

at the DOS prompt. PUSH.BAT will add that directory to its stack. If you try to save more than eight directories, PUSH.BAT will give you a "Stack is full" message.

To return to a previously "pushed" directory, simply type:

```
POP
```

Program Listing 1. PUSH.BAT saves the names of up to eight recently-visited directories, permitting you to recall those names later.

```
@ECHO OFF
REM PUSH.BAT
IF "%1"=="" GOTO NEW
IF "%9"=="" GOTO PUSHIT
ECHO Stack is full.
GOTO END
:PUSHIT
ECHO Pushing...
@ECHO OFF
SET POPER=POP
ECHO %0 %1 %HOMEDIR% %2 %3 %4 %5 %6 %7 %8 >
C:\BATCH\STK1.BAT
ECHO %POPER% %1 %HOMEDIR% %2 %3 %4 %5 %6 %7 %8 >
C:\BATCH\STK2.BAT
GOTO END
:NEW
DIRECTOR
:END
```

End

Program Listing 2. POP.BAT lets you quickly return to the eight directories most recently saved by PUSH.BAT.

```
@ECHO OFF
REM POP.BAT
IF "%1"=="" GOTO NEW
FOR %%X in (1,L) DO IF %%X==%1 GOTO SHOWEM
IF NOT "%2"=="" GOTO POPIT
ECHO Stack is empty.
GOTO END
:SHOWEM
TYPE C:\BATCH\STK2.BAT
GOTO END
:POPIT
ECHO.
ECHO Popping...
@ECHO OFF
SET POPER=PUSH
ECHO %0 %1 %3 %4 %5 %6 %7 %8 %9 > C:\BATCH\STK2.BAT
ECHO %POPER% %1 %3 %4 %5 %6 %7 %8 %9 >
C:\BATCH\STK1.BAT
%2\
CD %2
GOTO END
:NEW
C:\BATCH\STK2.BAT
:END
```

End

at the DOS prompt. POP.BAT will find the most recent entry placed on the stack by PUSH.BAT and switch you to that directory. Each successive use of POP will switch you to the next most-recent entry on the stack until no more entries remain, at which point you'll receive a "Stack is empty" message. To look at the list of directories on the stack at any time, type:

```
POP L
```

DIRECTOR.BAT is a modification of the batch file from

Program Listing 3. DIRECTOR.BAT is used by PUSH.BAT to capture the name of the current directory. It is a modified version of a batch file published in *DOS Resource Guide* #1 (page 75).

```
REM DIRECTOR.BAT
IF "%1"=="of" GOTO ICALLME
CTTY NUL
DIR DUMMY.$$$ | FIND "Directory of " > TEMP.BAT
CTTY CON
TEMP
GOTO END
:ICALLME
SET HOMEDIR=%2
DEL TEMP.BAT
C:\BATCH\STK1.BAT
:END
```

End

Program Listing 4. STK1.BAT is one of two files used by PUSH.BAT and POP.BAT to store a list or "stack" of directories. Both PUSH.BAT and POP.BAT modify this file, but this is its initial form.

```
REM STK1.BAT
PUSH *
```

End

DIRECTOR.BAT is a modification of the batch file from the "Home Please" tip mentioned above. DIRECTOR.BAT is called by PUSH.BAT to determine the name of the current directory, so it can be saved to the stack. STK1.BAT and STK2.BAT are batch files that contain the directory stack information. Both files are modified when you run PUSH.BAT and POP.BAT, so don't be surprised if you examine them and find that their contents are no longer the same as what is listed here.

Program Listing 5. STK2.BAT is the second file used to store directory information. Like STK1.BAT, it is modified by PUSH.BAT and POP.BAT.

```
REM STK2.BAT
POP *
```

End

PUSH.BAT, POP.BAT, and DIRECTOR.BAT must be placed in a directory listed in your PATH command. STK1.BAT and STK2.BAT may reside in any directory you choose, as long as you modify PUSH.BAT, POP.BAT, and DIRECTOR.BAT accordingly. I put STK1.BAT and STK2.BAT in my C:\BATCH directory. If you use a different directory, you'll need to change the pathname provided in lines 11 and 12 of PUSH.BAT; lines 9, 16, 17, and 22 of POP.BAT; and line 11 of DIRECTOR.BAT.

Herbert Feldman
Flower Mound, Texas

Screens in Living Color

Since reading the tip "Colorful Startup" (*DRG* #1, page 7), I have been toying with various PROMPT configurations in an attempt to improve on the black-and-white C:\> and C:\DOS> prompts. I've come up with a few suggestions that I think will be of interest to all your readers.

SCREEN COLOR VALUES

Foreground	Background
30 Black	40 Black
31 Red	41 Red
32 Green	42 Green
33 Yellow	43 Yellow
34 Blue	44 Blue
35 Magenta	45 Magenta
36 Cyan	46 Cyan
37 White	47 White

My three variations on this theme each contain three lines you can add to your AUTOEXEC.BAT file. They not only let you create a more colorful screen, but display vital information in the DOS prompt.

To display a white-on-blue screen and list the current directory in the DOS prompt, add these lines to your AUTOEXEC.BAT file:

```
ECHO ON
PROMPT $E[1;37;44m $P $G
ECHO OFF
```

To add the time, date, and your system's DOS version, substitute the following line for the second line shown above:

```
PROMPT $E[1;37;44m $T $D $V $P $G
```

To vary the foreground and background colors on the screen, substitute a new foreground color value (see the table "Screen Color Values") for 37 and a new background value for 44. For instance, the sequence to display magenta lettering on a white background is:

```
PROMPT $E[1;35;47m $T $D $V $P $G
```

All the above examples require that your CONFIG.SYS file contain a line such as:

```
DEVICE=C:\DOS\ANSI.SYS
```

that installs DOS's ANSI.SYS program.

Kenneth E. Vincent
San Antonio, Texas

Color Trouble

In the article "ANSI.SYS: A DOS Treasure," (*DRG* #3, page 48) you told me how to use an ANSI.SYS command to change my MS-DOS display. Your example, bright white text on a blue background, was exactly what I wanted on my screen, so I set about to create the required environment.

Unfortunately, none of your suggestions for entering the escape-character sequence required at the beginning of ANSI.SYS commands worked for me. The text editors on my machine would not put an escape character in a text file, and PROMPT ran out of environment space.

My solution, and suggested DOS tip, is a Basic program called SETSCRN.BAS that creates the text file containing the required escape character and character

Program Listing 7. SETSCRN.BAS creates a file to change the screen colors to bright white text on a blue background. This program works with both GW-Basic and QBasic.

```
10 OPEN "setscrn.txt" FOR OUTPUT AS 1
20 A$ = CHR$(27)+"[1;37;44m
30 WRITE #1,A$
```

End

string (see Listing 8). After running the program, I added the following line:

```
TYPE SETSCRN.TXT
```

to my AUTOEXEC.BAT. This solution works great!

I relish *DOS Resource Guide* Number 3. It is an excellent MS-DOS guide for the beginner and the advanced user. I have both 25 years of data processing experience and one year of MS-DOS experience. This is the best $6 I've ever spent on a manual.

Consider my second program, written in GW-Basic, a bonus tip. The program, SAVER.BAS, generates a simple but visually-interesting screen-saver display consisting of a yellow line that "bounces" around the screen.

David Ferrier
Edmonton, Alberta

Program Listing 8. SAVER.BAS creates a simple screen saver consisting of a yellow line that bounces around your display.

```
100 ' Moving line screen saver using GW-Basic graphics
110 ' D. Ferrier 1207 9837 110 St. Edmonton Alberta T5K 2L8
120 '————INITIALIZE LINE————
130 SCREEN 9: CLS: COL = 14
140 X1DIFF = -2 : Y1DIFF = 5 : X2DIFF = -3 : Y2DIFF = 4
150 X1=50: Y1=100: X2=200: Y2=300
160 '————DISPLAY LINE————
170 WHILE 1=1
180 X1 = X1+X1DIFF: Y1 = Y1+Y1DIFF
190 X2 = X2+X2DIFF: Y2 = Y2+Y2DIFF
200 LINE (X1V,Y1V)-(X2V,Y2V), 0
210 LINE (X1,Y1)-(X2,Y2), COL
220 X1V = X1: Y1V = Y1: X2V = X2: Y2V = Y2
230 IF X1 > 640 OR X1 < 1 THEN X1DIFF = X1DIFF*-1
240 IF Y1 > 350 OR Y1 < 1 THEN Y1DIFF = Y1DIFF*-1
250 IF X2 > 640 OR X2 < 1 THEN X2DIFF = X2DIFF*-1
260 IF Y2 > 350 OR Y2 < 1 THEN Y2DIFF = Y2DIFF*-1
270 IF INKEY$ <> "" THEN SYSTEM
280 WEND
```

End

Mr. Ferrier's SETSCRN.BAS program works fine with either GW-Basic or DOS 5.0's QBasic. With either language, when you type in the program and run it, the file SETSCRN.TXT will be created. If you keep SETSCRN.TXT in your root directory, just add the lines:

```
TYPE SETSCRN.TXT
CLS
```

to the end of your AUTOEXEC.BAT file. If you wish to keep SETSCRN.TXT in a different directory, such as C:\UTILS, you'll need to include the complete pathname when you invoke TYPE:

```
TYPE C:\UTILS\SETSCRN.TXT
CLS
```

The CLS command clears the screen to a blue background with a white DOS prompt in the upper lefthand corner.

The other program, SAVER.BAS, also works fine with either GW-Basic or QBasic. It is a simple screen-saver you can use when you wish to walk away and leave your monitor turned on for a period of time but would like a continuously changing screen display in your absence.

First, start your version of Basic, type in the listing, save the listing as SAVER.BAS, and run it. You'll see a bouncing yellow line on the screen until you press a key. Now that you know SAVER.BAS is working, create a short batch file called SAVE.BAT to make it easier to activate the program.

If you're running GW-Basic and both GWBASIC.EXE and SAVER.BAS are located in your C:\DOS directory, SAVE.BAT looks like this:

```
@ECHO OFF
CD\DOS
GWBASIC SAVER
```

A version of SAVE.BAT for QBasic users might look like this:

```
@ECHO OFF
CD\DOS
QBASIC /RUN SAVER
```

Be sure SAVE.BAT is in a directory that is included in your PATH statement. Then, any time you are at the DOS prompt and you want to activate the bouncing line display, simply type:

```
SAVE
```

and press the Enter key.

—eds.

Better Directory Assistance

I recently read a tip in your magazine entitled "Directory Assistant" (*DRG* #3, page 29), which describes a batch file called GO.BAT. I was interested in this utility because, even though I usually use a disk manager, I sometimes find myself at the C:> prompt, where I'm forced to manually maneuver around my computer's five disk drives. I thought GO.BAT was an excellent idea but found it somewhat limiting (see the Program Listing).

To begin with, GO.BAT doesn't let you run programs on drives other than the one on which it resides. In addition, it won't let you enter command-line parameters. I ran into these limitations quite frequently, so I made a few changes to GO.BAT and renamed it RUN.BAT.

First, I added error messages. The initial IF statement checks to ensure that you did indeed give RUN.BAT a parameter to work with. The second checks to see if you mistakenly tried to enter a command-line parameter without a program to run. (I'll explain how to use command-line parameters with RUN.BAT in a moment.)

I added a FOR statement for each drive installed in my computer. Each one checks for a drive change; if none is found, the program skips to the label LOOP. If a drive change is found, the program goes to the label DRIVE.

After RUN.BAT changes drives, it uses a SHIFT statement to shift the command-line parameters, so that other sections of the batch file will operate correctly whether or not a drive is specified on RUN.BAT's command line. The remainder of RUN.BAT closely parallels GO.BAT.

To permit passing command-line parameters to the program you wish to run, I had to provide a means for RUN.BAT to distinguish between parameters and directories or programs. My solution was to precede command-line parameters with a plus sign (+). The first IF statement after the label LOOP checks to ensure that the first parameter is not a +. If it is, an error exists because a + in the RUN.BAT command line must be followed by the name of an executable file. Just as with

Program Listing. RUN.BAT lets you use a single command line to run a program on any drive or in any directory.

```
@ECHO OFF
IF “%1"=="" GOTO ERROR
IF “%1"=="+" GOTO ERROR1
CD\
IF “%2"=="" GOTO LOOP
FOR %%P IN (A: a:) DO IF “%1"=="%%P" GOTO
DRIVE
FOR %%P IN (B: b:) DO IF “%1"=="%%P" GOTO
DRIVE
FOR %%P IN (C: c:) DO IF “%1"=="%%P" GOTO
DRIVE
FOR %%P IN (D: d:) DO IF “%1"=="%%P" GOTO
DRIVE
FOR %%P IN (N: n:) DO IF “%1"=="%%P" GOTO
DRIVE
GOTO LOOP
:DRIVE
   % 1
   SHIFT
   CD\
:LOOP
   IF “%1"=="+" GOTO ERROR1
   IF “%2"=="" GOTO DONE
   IF “%2"=="+" GOTO DONE
   IF EXIST %1\NUL GOTO CHGDR
   GOTO ERROR3
:CHGDR
   CD %1
   SHIFT
   GOTO LOOP
:DONE
   IF EXIST %1.COM GOTO RUN
   IF EXIST %1.EXE GOTO RUN
   IF EXIST %1.BAT GOTO RUN
   IF EXIST %1\NUL GOTO CHGDR1
:CHGDR1
   CD %1
   IF EXIST %1.COM GOTO RUN
   IF EXIST %1.EXE GOTO RUN
   IF EXIST %1.BAT GOTO RUN
   SHIFT
   IF “%1"=="" GOTO ERROR2
:RUN
   IF “%2"=="+" SET CMD=%1 %3
   IF “%2"=="" SET CMD=%1
   % C M D %
   GOTO END
:ERROR
   CLS
   ECHO You must enter Drive Directory
Program +
   ECHO Parameters
   GOTO ENDERROR
:ERROR1
   CLS
   ECHO You can enter the + before
   ECHO command line parameters only
   GOTO ENDERROR
:ERROR2
   CLS
   ECHO The executable file you have entered
   ECHO does not exist
   GOTO ENDERROR
:ERROR3
   CLS
   ECHO You have selected an invalid directo-
ry
:ENDERROR
```

GO.BAT, if you don't list a second parameter, RUN.BAT can begin to process your request. If the second parameter is a +, a command-line parameter exists, and the request is processed.

The DONE label works in the same way as it does in GO.BAT, simply checking to ensure that the subdirectory or program exists. The main difference here is that the last line of the DONE section checks to see if the NUL file exists in the subdirectory, which is my way of finding out if the subdirectory exists. (A quirk of DOS is that each subdirectory contains an invisible entry called NUL. So, if you test for the existence of the NUL entry and it does not exist, then the subdirectory doesn't exist.)

Once RUN.BAT determines that the subdirectory or executable file exists, then it skips to the RUN label. RUN functions essentially as it does in GO.BAT. The only difference is that I have added IF statements to check for a command-line parameter. If none exists, then CMD is set to the program name. If a command-line parameter does exist (if the command line includes a +), CMD is set to the program name followed by a space and the specified command-line parameter. Remember, when you supply a command-line parameter to the program you wish to run using RUN.BAT, the variable %2 will be a +, so CMD must be set to %1 (the program name) and %3 (the command-line parameter), with a space between the two.

The batch file concludes by executing the commands after the END label. First, I set CMD equal to nothing, which deletes it from the DOS environment. Then, I include commands to return to the root directory of drive C and clear the screen.

Using RUN.BAT is quite simple. If, for instance, you want to execute QBasic, which resides on drive D, and load a program called MYFILE.BAS, you type:

```
RUN D: QBASIC + MYFILE.BAS
```

I hope you find RUN.BAT as useful as I have.

John F. Lohmann
Havre, Mont.

Did you ever notice that C:>doesn't really point anywhere?

If you need DOS direction, get into the only quarterly devoted to its inner workings: DOS RESOURCE GUIDE. Each issue is packed with shortcuts, how-tos, and solid explanations of DOS's most intimate workings.

DOS RESOURCE GUIDE #1

Make DOS Work for You

Covers six areas of computing that are vital to DOS users: start-up files (AUTOEXEC.BAT and CONFIG.SYS), batch file creation, hard drives, safe backups, organizing file management, and hardware maintenance. Includes: List'em, a batch file that helps you keep appointments; 10 tips to keep your hard drive healthy; speed-up tips for a logy hard drive; a batch file that automates backups; DOS shortcuts, hardware dos and don'ts. Plus 42 provocative pointers on DOS.

DOS RESOURCE GUIDE #2

Guide to Windows Software

SOLD OUT

A compendium of software briefs that [illegible] what's up in the world of Windows. (Even a DOS user has to keep up with the competition.) Guide #2 briefly reviews over 140 programs for Microsoft Windows, including word processors, communications programs, financial and mathematical programs, management programs, developer's tools, and graphics software. Complete list of manufacturers.

DOS RESOURCE GUIDE #3

The Essential Guide to DOS

Who are we to argue with success! Our second edition of this classic is updated to include information on DOS 5.0. Guide #3 includes an annotated list of DOS files and when to use them; articles that will change your view of CHKDSK, COPY, and XCOPY; a cluster of batch file assists; and some clear-headed advice on organizing files on your hard drive. Plus 14 titillating tips on DOS.

DOS RESOURCE GUIDE #4

The Complete DOS Q&A

Breaks DOS into functional sections: Fundamentals, Setup, Improving Productivity, Memory Management, Hardware Upgrades, Batch Files, and Using Windows. Each section is followed by a question and answer session on the topic. Plus, seven short programs to protect your data; do-it-yourself memory, advice on CONFIG.SYS and AUTOEXEC.BAT, and beginner's Basic. Not to mention nine tips for the DOS timid.

ORDER YOUR BACK ISSUES TODAY!

Each back issue of *DOS RESOURCE GUIDE* is $5.50 postage paid.
Call 1-800-238-6580 (9AM to 5PM EST).

Or Write: DOS RESOURCE GUIDE
P.O. Box 3162
Salisbury, MD 21802

Develop Your PC's Personality

Fine-tuning CONFIG.SYS and AUTOEXEC.BAT is one of the best ways to get your computer to behave the way you think it should.

by Hardin Brothers

You probably think that your computer is much like every other. Maybe you've installed more memory than the guy across the hall, and maybe you have a larger monitor than your nephew, Trevor, but you probably could move your business inventory, your personal financial records, and your daughter's soccer team schedule to any other computer and soon feel at home. After all, computers are interchangeable commodities, and an MS-DOS computer is much like any other machine.

Every IBM compatible does have its own personality, however. The commands available from the C:> prompt vary according to the version of DOS you use, and the way you store files is determined by the subdirectory structure that you create on your disk. And although you may never have looked at them, your hard disk contains two files that re-create your computer's personality every time you boot up. No matter what version of DOS you use, these two personality files are called CONFIG.SYS and AUTOEXEC.BAT.

CONFIG.SYS: A text file containing system setup information that DOS uses each time you boot up your computer.

Program Type: External.

Available: MS-DOS 2.0 and later.

AUTOEXEC.BAT: A special batch file that provides a means of automatically issuing a series of commands whenever you start up your computer.

Program Type: External.

Available: MS-DOS 1.25 and later.

Many users these days never look at CONFIG.SYS or AUTOEXEC.BAT. Starting with Version 4.0, DOS creates or modifies these files for you as it installs itself on your computer. And most application programs now make any necessary changes to these files during the programs' automated installation procedures.

But you should think about taking a closer look at them and maybe doing a little fine-tuning. It's a simple process to manually modify both CONFIG.SYS and AUTOEXEC.BAT to make your computer behave the way you want. And when you are finished, your computer may run faster, better suit your way of working, and have more memory accessible for large applications.

How CONFIG.SYS And AUTOEXEC.BAT Work

When you turn on your computer, program code in the machine's built-in memory, ROM, sets up your hardware, prepares it to get to work, and then loads the bootstrap routines that, in turn, load and configure DOS.

During this process, called initialization, DOS reads your CONFIG.SYS file. Some of the instructions in that file tell DOS to set aside blocks of memory for file operations, while other instructions load special driver programs that make your hardware run efficiently.

When DOS is finished configuring itself, it loads COMMAND.COM, the command processor that displays the DOS prompt on the screen and reacts to the commands that you type at the prompt. But before it lets you type anything, COMMAND.COM runs the commands in your AUTOEXEC.BAT file just as if you had typed them in from the keyboard.

You can't prevent DOS from following the commands in CONFIG.SYS, because it reads this file before it looks at the keyboard. And usually you don't want to stop DOS from executing these commands. (Sometimes, if you are lucky and your timing is exact, you can stop the processing of your AUTOEXEC.BAT file by pressing Ctrl-Break at just the right moment.) It is possible, however, for errors in either file to keep your computer from successfully booting up, and if your computer does lock up during its bootup cycle, you've got something of a problem. Pressing Ctrl-Alt-Del or the reset button won't solve anything, because DOS simply reboots and makes the same mistakes again.

For this reason, you must proceed cautiously when making any changes to your CONFIG.SYS and AUTOEXEC.BAT files. And before you start, you should have an emergency bootup floppy disk on hand as insurance. To get started, put a new disk in drive A and type:

```
FORMAT A: /S
```

The /S tells the FORMAT command that you want to make a system disk containing DOS and COMMAND.COM. (Some older versions of DOS require that you manually copy COMMAND.COM to the bootup floppy after the formatting process ends.)

After formatting the floppy disk, use the following two commands to put a backup copy of your configuration files onto this disk as references:

```
COPY C:\CONFIG.SYS A:\CONFIG.BAK
COPY C:\AUTOEXEC.BAT A:\AUTOEXEC.BAK
```

If you have special hardware that must be initialized with CONFIG.SYS commands before your computer will work (for example, if you use a disk-compression program such as Stac Electronics' Stacker or special hard disk partitioning software), create a CONFIG.SYS file on your bootup floppy disk that contains only those commands.

Finally, copy a simple text editor to the floppy disk. If you use DOS 5.0, the EDIT utility is a good choice. (Be sure, however, that you copy EDIT.COM, EDIT.HLP, and QBASIC.EXE., because EDIT won't work without QBASIC.EXE.) With older versions of DOS, you can use the archaic EDLIN or another text editor.

Put the floppy disk in drive A and reboot your computer; it should boot up correctly. If not, you probably have omitted something important from the CONFIG.SYS file on the floppy disk. Fix the file, put the disk back in the drive, and try to reboot. When you are satisfied that the floppy disk will let you boot up your computer, label it as an emergency bootup disk and put it safely away. You can use it any time you make a mistake in one of your configuration files to get your computer to boot up correctly.

Making File Changes

Both CONFIG.SYS and AUTOEXEC.BAT are plain text files. If you create or modify them with your word processor, make sure you save them in ASCII (text) or DOS format.

You can create either file directly from the DOS prompt with a command such as:

```
COPY CON CONFIG.SYS
```

which puts anything you type into the CONFIG.SYS file. To close the file and return the computer to normal operation, press Ctrl-Z and Enter at the beginning of a new line. Be careful, though. If CONFIG.SYS already exists, this method will wipe out the file's contents, substituting the new information you type.

If you simply want to add a line or two to one of these files without changing anything that the file already contains, use a command such as:

```
COPY AUTOEXEC.BAT + CON
```

Again, press Ctrl-Z and then Enter to stop adding new lines to the file and get back to normal operation. This procedure appends the

TIP DOING WHAT DOS DOESN'T

Macros created with DOS 5.0's DOSKEY utility override standard DOS commands that have the same name. You can, therefore, use DOSKEY to modify the way in which DOS executes a specific command.

Suppose, for instance, that you always want the DOS MEM command to display information regarding any high memory (RAM above 640K) that your system contains. The following command would create a DOSKEY macro that automatically adds the "classify" parameter, /C, to the MEM command each time it is run:

```
DOSKEY MEM=MEM /C
```

Thereafter, whenever you type MEM, DOS will display the size of each program in high memory, provide a summary of high memory use, and list the largest unused memory blocks in high memory.

This technique works with most standard DOS commands.

—Jack Nimersheim

new commands to the existing file.

Although both of the techniques explained above let you enter individual lines in an ASCII file and make changes to each line (before pressing Enter) by backspacing and retyping the line, neither lets you change lines that already exist in the file. To do that, you need a text editor or a word processor (just make certain you save your changes in ASCII format).

If you simply want to view either file, you can issue the TYPE command. If the file is too long to display on the screen at once, use a command such as:

```
TYPE AUTOEXEC.BAT | MORE
```

Fine-Tuning CONFIG.SYS

CONFIG.SYS tends to be small; rarely will it contain more than a single screenful of commands. Although you can include many different commands in a CONFIG.SYS file, few people use more than a handful.

Any line that begins with the word REM and a space is a remark, which means DOS ignores the rest of the line as it processes the file. You can use remarks to leave notes to yourself about commands in the file. In early versions of DOS, an error message appears on the screen when a remark in the CONFIG.SYS file is processed. Even though the message and error are harmless, this bug was fixed in DOS 3.2.

All other lines in CONFIG.SYS begin with a command or keyword (see below) that is followed by an equal sign and information for the command to process. It is best not to leave any spaces on either side of the equal sign, although spaces usually won't cause problems.

The most common keywords are FILES and BUFFERS. The FILES command tells DOS how many files may be open at one time. Most word processors, databases, spreadsheets, and CAD packages specify a minimum number for the FILES setting. Use the largest number demanded by any of your programs. If you aren't sure, try:

```
FILES=20
```

This will satisfy most programs.

If you use pop-up utilities (such as Borland International's Sidekick) that employ disk files, increase the number of files by one or two. If you set FILES too low, one of your application programs will report that it can't open its files and probably will stop operating. DOS reserves memory for each file, thereby reducing the memory available for use by your applications. So, don't set FILES unnecessarily high.

The BUFFERS setting tells DOS to reserve some room for disk buffers. These buffers are used when a program reads part of a disk sector and then, at a later time, reads more of the same sector. DOS keeps partially used sectors in memory so that it can supply the second request without stopping to get more data from the disk.

If you set BUFFERS too low, your computer will spend more time reading the disk and less time doing your work. But if you set BUFFERS too high, your computer will spend too much time searching for information in the list of buffers before it goes out to read more data from disk. You'll have to experiment with the BUFFERS setting to see which value makes your computer run most efficiently. Each buffer takes just over 512 bytes of memory, so a high BUFFERS setting could possibly use so much memory that your largest applications won't have sufficient memory to run.

Assuming that you have at least 640K of RAM installed in your computer, a good rule of thumb is to set BUFFERS between 20 and 40 if you use a large database or large accounting program. Set BUFFERS to between 10 and 20 for other programs. But, again, experimentation may help you find the optimum setting.

The other common CONFIG.SYS commands are DEVICE and INSTALL. Both of these commands load special drivers that control your computer's hardware or hardware-like devices. Most versions of DOS include a file called ANSI.SYS, for example, which lets you define colors for the DOS screen and redefine keys on the keyboard. Many versions of DOS also include RAM disks and other utilities that may be installed in CONFIG.SYS. In addi-

TIP MORE WORK FROM ONE

If you need to fit more than 127 characters on the command line—DOS's limit— you're in luck. DOS provides a simple way for doing this; the trick comes in especially handy if, for example, you want to copy a great many files and combine them into a single file. The technique involves organizing your work using a standard set of file extensions. You then can use DOS's FOR command and wildcards to copy and combine dozens or even hundreds of files.

For example, if *.DAT, .TXT,* .DOC will find all the files—and no others—that you wish to combine and copy to NEWWORK.ALL, you could type the following on the DOS command line:

```
FOR %F IN (*.DAT *.TXT *.DOC) DO COPY
  NEWWORK.ALL+%F NEWWORK.ALL
```

If you want to automate this process in a batch file, you might try this command sequence:

```
FOR %%F IN (*.DAT *.TXT *.DOC) DO COPY
  NEWWORK.ALL+%%F NEWWORK.ALL
```

—Legare Coleman

tion, memory management software, such as Quarterdeck Office System's QEMM386 and Qualitas's 386MAX, is installed with similar commands, as are many hard disk partitioning and compression programs.

Your DOS manual explains other commands that can be included in your CONFIG.SYS files. Some, such as BREAK, are rarely used at all, while others, such as COUNTRY.SYS, are used to give DOS an international flavor. You might want to look through your DOS manual to see if any of these commands or device drivers will help you get your work done more efficiently.

Customizing AUTOEXEC.BAT

Any batch file, including AUTOEXEC.BAT, is mostly a collection of commands that you can type at the DOS prompt. The purpose of batch files is to eliminate repetitive typing. AUTOEXEC.BAT, for instance, removes the need for you to type the same series of commands every time you boot up your computer.

Some AUTOEXEC.BAT files are quite long. You can call on AUTOEXEC.BAT to customize the DOS prompt using the PROMPT command, to create environment variables for the applications you run with the SET command, to send commands to ANSI.SYS to redefine keys and change the color of the screen, to set up DOS's search path for finding executable files, and to install pop-up utilities that you can call from within other applications.

You can do several things to make AUTOEXEC.BAT easier to understand and modify. First, use a lot of remark lines; they let you add reminders that say why you have included each group of lines.

Also, group the commands in your AUTOEXEC.BAT file logically. For example, if you need four lines to send commands to ANSI.SYS and to set your DOS prompt, then put them all in one place. If AUTOEXEC.BAT is going to install three pop-up utilities, put the installation commands together. If you need 10 lines to create environment variables with the SET command, put those together.

Finally, use environment variables to temporarily store information. For instance, I use several different AUTOEXEC.BAT files, each customized to one of the many ways I use my computer. Suppose that in one of these files I need to change the PATH setting to correctly install a pop-up program (which I'll call POPTART), but I don't want to forget my current PATH setting because I want to use it again later. The following segment of my AUTOEXEC.BAT lets me use a new setting without losing the old setting:

```
REM Save the PATH:
SET OLDPATH=%PATH%
REM Reset the PATH for POPTART:
PATH C:\POPUP;C:\POPUP\DATA
REM Install POPTART:
POPTART
REM Recover original PATH
PATH %OLDPATH%
SET OLDPATH=
```

Several lines in this section work because of the way environment variables are used in batch files. You can see the current variables by typing SET at the DOS prompt, and you can create a new variable with the SET command. For example:

```
SET TEMP=E:\TMP
```

creates a variable called TEMP and gives it the value E:\TMP. Now, if you type SET again, you'll see this variable added to the list.

If you include the name of an environment variable in a batch file and place the name inside percent signs, the name is replaced with everything after the equal sign (if you used the SET command). In my example, DOS would replace any occurence of %TEMP% with E:\TMP.

The example above illustrates how a variable called OLDPATH is created and how the current contents of the PATH setting is stored there: The PATH is modified, the POPTART program is installed, and the PATH is restored to its original value. Finally, the OLDPATH variable is erased so that it won't take up unnecessary space in the environment.

Tricks such as this can make your AUTOEXEC.BAT file much more productive. Feel free to experiment with both your configuration files. As long as you have a copy of the original files and a bootup floppy disk, you can't accidentally hurt anything. Try tricks that you have read about in the *DOS Resource Guide* and in other magazines, tips you pick up at user's group meetings, and new ideas of your own. You'll soon have your computer working for you and acting the way that you want it to, instead of being forced to work the way it demands. ■

Free-lance writer Hardin Brothers has been working with and writing about computers for 13 years.

TIP INFO AT YOUR FINGERTIPS

It took four major upgrades, but Microsoft finally added an on-line help feature to DOS. DOS 5.0's HELP.EXE utility lets you request information on most DOS operations. The following command, for instance, displays an on-screen description of the COPY command, along with its syntax and valid parameters:

```
HELP COPY
```

To request information regarding any other DOS 5.0 command, simply substitute that command for *COPY*.

—J.N.

Positive Attributes

Need to back up your disk, protect files from casual snooping, and copy a group of unrelated files? ATTRIB can help.

by James Langdell

When you invoke the DIR command, DOS normally doesn't tell you everything about your files. Each file has a few secrets that don't appear in a directory listing. These secret facts, known as file attributes, have an impact on how secure the files on your computer are.

File attributes determine if a file can be deleted or changed, indicate whether it was backed up recently, and control whether a file's name appears in the directory listing. Once you know how to identify and change file attributes, you can take advantage of them to prevent important files from being deleted or changed unintentionally and to back up your files more efficiently to preserve your latest creations.

Defining File Attributes

A file's attributes are stored in one of the 32 bytes devoted to its directory listing. In addition to the attribute byte, a file's directory listing also stores the file's name, the time and date when it was last changed, the starting location of the file on the disk, and the size of the file. Within the attribute byte, each file attribute—read-only status, for example—is defined by one bit of the byte's eight bits. When a value of 1 is stored in a bit, that attribute is turned on; a zero indicates that the attribute is turned off.

ATTRIB: Displays or alters the attributes of DOS files. It is used most frequently to protect files from accidental erasure by making them read-only, which means you may not overwrite them. DOS 5.0 extends DOS's attribute controls, letting you hide and unhide files from prying eyes.

Command Type: External

Available: MS-DOS 3.0 and later. Enhanced in Version 3.2 and again in Version 5.0.

The ATTRIB command gives you a way to reveal and revise the attributes of each file on your computer. In early DOS versions, the command could control only a file's read-only attribute. In MS-DOS 3.2, Microsoft added an archive attribute. MS-DOS 5.0 lets you control two additional attributes: the hidden attribute and the system attribute.

If you type in the ATTRIB command by itself at the DOS prompt, you receive an on-screen report listing the attribute status of files in the current directory. To the left of each filename, a letter appears for each file attribute that is turned on.

When the read-only attribute (indicated by the letter *R*) appears in the on-screen report, read-only status is in force. This means that you may not change or delete that file with MS-DOS commands such as DEL and REN or edit it with Edlin. This protection doesn't make the file untouchable, however. The FORMAT command, the equivalent of kryptonite for read-only files, will reformat a disk, wiping out all files and directories on it—regardless of their attributes.

The archive attribute (indicated by an *A*) helps you keep track of whether a file was created or modified since the last time you backed up your files. Knowing this information lets you back up your files more efficiently.

The hidden attribute (indicated by an *H*) prohibits a file from being displayed by the DIR command (except under special conditions in MS-DOS 5.0). For example, because Microsoft marks the two MS-DOS boot files IO.SYS and MSDOS.SYS (IBMBIO.COM and IBMDOS.COM in PC-DOS) as hidden, they do not appear when you take a directory of the root directory. Hiding files makes them less vulnerable to unauthorized scrutiny and modification.

The system attribute (indicated by an *S*) marks a file as a central part of the operating system. Probably the only files on your computer with this attribute are the two hidden boot files. (The system attribute is not, however, turned on for COMMAND.COM, the third major file in DOS.)

So far, I've mentioned four file attributes: read-only, archive, hidden, and system. But, since eight bits are available in each file's attribute byte, you well may wonder what the other four bits mean. Two of the bits are used to identify an item as a subdirectory or a drive's volume label. Microsoft hasn't assigned a function to the remaining two bits. Of the four attribute bits, the subdirectory attribute is the only one that plays a role in the processes described in this article.

Attributes and the DIR Command

In MS-DOS 5.0, the DIR command can display filenames and list all four types of attributes, or it can display only files with chosen attributes. The DIR command's default is to show all files except hidden files, but you can convince it to show the hidden files by adding the attribute switch /A. The proper command for doing this is:

```
DIR /A
```

(If you enter the command `DIR /AH` instead, DIR displays a list of only the hidden files.)

By defining an environment variable named DIRCMD, you can change the DIR command's default setting so it always displays hidden files. The SET command is the tool you use to define DIRCMD:

```
SET DIRCMD=/A
```

You also can use the SET command by itself to see all the environment variables (such as the PATH definition) in effect.

The /A switch of DOS 5.0's enhanced DIR command makes it easy to display subdirectories only. In past DOS versions, you had to use a command such as:

```
DIR | FIND "<DIR>"
```

to filter the DIR command's output to isolate the subdirectories.

With DOS 5.0, you can type the following:

```
DIR /AD
```

The *A* stands for *attributes*, and the *D* stands for *directories*.

(If you use DRDOS 6.0 rather than Microsoft's DOS 5.0, the internal DIR command does not have the attribute-related features described here. However, these features are part of DRDOS's external XDIR command.)

Protecting Files with Attributes

Turning on the read-only attribute of a file is a way to protect against accidentally changing or deleting the file. You can use this feature to protect the master templates for forms that you fill in regularly using a word processor. If all the files for master forms have the extension .FRM, you can give them all read-only protection with the following command:

```
ATTRIB +R *.FRM
```

You may copy these read-only .FRM files and then change the copies, but you may not change or delete the originals while the read-only attribute for those files is turned on.

In the ATTRIB command, the plus sign (+) tells DOS to turn on a

BACKING UP VS. ARCHIVING

Now that you know about some DOS tools that can help you back up your files, think about what you're trying to accomplish by duplicating the files regularly. Even though a file's archive attribute may be used to carry out a BACKUP command, archiving and backing up are different exercises.

You back up hard disks to protect their contents against unexpected attacks on your system—from a virus, accidental deletion or corruption of files, or physical damage to your computer. The goal is to restore your hard disk to its original condition as quickly and completely as possible. The more carefully considered your backup strategy, the less valuable data and work time you'll lose, regardless of when disaster strikes.

When you get down to it, the best backup method is any one that you actually use. In DOS, the BACKUP and RESTORE commands have changed greatly from version to version, becoming progressively more useful and flexible. However, third-party backup programs offer other features, such as file compression, that speed the process or save more disk space than DOS's BACKUP program. If you find a program with features that increase your motivation to back up your files regularly, that alone may be reason enough to purchase the program.

Guarding against unpredictable calamities isn't the only rationale for backing up a disk. You might, for example, upgrade you computer system and need to transfer important files to the new machine, need to replace the hard disk on your present system, or decide to upgrade to a different operating system and have to format or partition *your disk* differently.

Whatever the reason, you'll want to capture your computer setup exactly but be able to reconstruct it on a system with slightly different hardware and software.

If you simply want to make a mirror image of your hard file's attribute; a minus sign (–) tells DOS to turn off that attribute. If you run the ATTRIB command by itself, you see entries such as these in the report it displays:

```
    MONTHLY.APR
  R MONTHLY.FRM
    MONTHLY.MAY
  R PURCHASE.FRM
  R TAX.FRM
```

Later, if you want to start using a revised version of one of the templates, you can turn off the read-only attribute of that template file. For example, to make it possible to change the PURCHASE.FRM template file, enter this command:

```
ATTRIB -R PURCHASE.FRM
```

But remember to turn the template file's read-only attribute back on after you've made the desired changes.

Although setting the read-only attribute of a file protects it against a good deal of harm, not all application programs pay attention to this attribute. And, while the attribute might provide some protection against viruses, it doesn't take much to create a virus program clever enough to bypass that such protective measures.

Hiding Files with Attributes

By setting the hidden attribute of a file, you can prevent other users from realizing that the file is on the disk. This is useful if a file contains data you don't want other users to read or change, or if you have programs that you don't want others to run. For example, if you want to keep a report on a product in development under wraps, you can hide that file (when you're not working on it) with a command such as:

```
ATTRIB +H DEATHRAY.DOC
```

The DEATHRAY.DOC file won't appear in the DIR command's reports, and programs that respect the hidden attribute won't give you access to a file protected in this way.

You also might hide command and program files that can be dangerous if used thoughtlessly or maliciously. Under certain circumstance, for instance, hiding FORMAT and RECOVER might be a good idea.

Making a file hidden, however, does not put it completely out of reach. Anyone who knows that a file exists can refer to it by name in some DOS commands (such as `TYPE DEATHRAY.DOC`), and DOS will grant the user access. If you hide a program file, it still will run if someone asks for it by name in the correct directory (or if the file is hidden in a directory mentioned in PATH or in the PATH statement of your AUTOEXEC.BAT). Of course, anyone who knows how to use the ATTRIB and DIR commands in DOS 5.0 can use them to locate and peruse a file almost as easily as you hid it. The main advantage of the hidden attribute is concealing sensitive files from casual view.

Sometimes an application program will set certain files' read-only or hidden attributes to shield those files from view and modification. If you have difficulty removing an old program from a directory, invoke ATTRIB to see if the directory contains any read-only files. You can make such files visible and removable by typing:

```
ATTRIB -H -R
```

Other Ways to Change Attributes

You may carry out many of the ATTRIB command's functions within the DOS shell. The procedure involves selecting files you want to change from the file list, bringing down the File menu, and choosing the Change Attributes option. The shell displays a dialog box that lets you change hidden, system, archiving, and read-only attributes.

A number of shell programs from

disk on the same disk or another disk after one of these operations, the BACKUP command is the most straightforward DOS command to use. If you plan to arrange your file differently on the new or modified machine, however, you might be better off using the XCOPY command to back up files and directories in groups, which you can rename or rearrange more flexibly on the new hard disk.

Archiving is different. In its purest sense, it is a way of storing files that you may need on occasion. You might, for instance, archive all the files for a completed project or preserve financial files from past years.

When you archive files, make sure you name the files and label the floppy disks or tapes clearly enough to identify them, perhaps years later. Also, keep in mind that you might need to use those archived files on a machine quite different from the one you're currently using. To assure that your archived files are still readable at that time, avoid using special file formats (such as compressed files) that might not be readable on another machine.

For purposes other than archiving, file compression can be a valuable way to increase the number of files your hard disk can hold. You might be able to reduce individual files by as much as 800 percent by compressing them. Data files usually compress more than program files, because compression software generally looks for repeated patterns in a file and indexes the patterns only once. Data files usually have more repetitious elements than programming code.

It's important to be able to tell if a file is compressed or if it's in a complete, ready-to-use form. Many compression schemes give compressed files a new extension, such as .ZIP. (The original filename is stored in the compressed file and is restored when you expand the compressed file.)

Unfortunately, no compression method is standard for all DOS computers. File attributes, as useful as they are in other ways, won't give you a clue about whether or not a file is compressed.

—J.L.

third-party software developers can alter file attributes, as well. One of the initial attractions of The Norton Utilities, for example, was that—long before DOS itself provided even a rudimentary ATTRIB command—it let users hide and unhide files by manipulating their attributes.

Using Archive Attributes

The archive attribute comes into play when you're ready to use the BACKUP or XCOPY command. Either of these commands lets you specify a parameter that duplicates only those files for which the archive attribute is turned on.

The primary use of the archive attribute is to identify files you have changed or added to your computer since you last backed up the contents of its disks (see the sidebar, "Backing Up Vs. Archiving" on page 16 for an explanation of the differences between the two procedures). You might, for instance, do a full backup of your hard disk onto a set of floppy disks using the following command:

```
BACKUP C:\ A: /S
```

The /S tells BACKUP to duplicate the contents of all subdirectories on drive C.

When the backup is complete, you can enter this command:

```
ATTRIB -A C:\ /S
```

to turn off the archive attribute throughout the hard disk. Now, imagine a few days have passed since the full backup. You can do an incremental backup, a much less extensive backup operation that saves only the files that were changed or added to your hard disk since the full backup. DOS will have turned on automatically the archive attribute of files you've added or changed during the past few days. You can tell the BACKUP command to copy just those files by entering this command:

```
BACKUP C:\ A: /M /S
```

In this command, /M switch tells BACKUP to copy files only if their archive attribute is turned on. As each file is copied, BACKUP turns off its archive attribute, indicating that this file doesn't need to be copied the next time you perform an incremental backup.

Since you're copying fewer files at a time during an incremental backup, it's faster than a full backup. The ease of incremental backups might inspire you to backup more frequently.

If you ever need to restore the contents of a hard disk that you have backed up incrementally, be sure to start by restoring the contents from the set of floppy disks that completely backed up the hard disk. Then, restore the contents of each incremental set of backup disks—in chronological order. This is the only way to ensure that the most recent version of a frequently revised file winds up on the fully restored disk. Keep in mind that any files you have deleted since the last full backup might still be present on the restored disk.

Copying Files Selectively

Like BACKUP, XCOPY recognizes the archive attribute. You can use XCOPY, instead of BACKUP, to incrementally back up files with a command such as this:

```
XCOPY C:\ A: /M /S
```

You also can use the archive attribute to plan which files you want to copy from a source disk—even when your actions are unrelated to backup or archiving strategies. Say, for example, that you want to copy only 30 of a directory's 40 text files. If the files you want to copy have names that distinguish them from other files in the directory (for example, if all the desired filenames begin with *R* or all have .BAK file extensions), you can use the wildcards with the COPY or XCOPY command. However, if no name pattern distinguishes the files you want to copy, you can turn off the archive attribute of the undesired files one by one, "tricking" XCOPY into not copying them.

Start by turning on the archive attribute for all files in the directory:

```
ATTRIB +A
```

Now enter commands that turn off that attribute for the files you don't want to copy:

```
ATTRIB -A  *.BIG
ATTRIB -A  REPORT.788
ATTRIB -A  RESUME*.*
```

You then are ready to copy onto a floppy disk the remaining 30 files, which still have their attribute turned on:

```
XCOPY C:*.* B: /A
```

The /A switch tells XCOPY to copy only files whose archive attribute is turned on. In this case, the archive attribute remains turned on for each file after it is copied.

If the files you want to copy won't fit on a single floppy disk, the archive attribute can help XCOPY keep track of the names of the files that DOS hasn't yet copied. In this case, enter this command:

```
XCOPY C:*.* B: /M
```

The /M parameter tells XCOPY to turn off the archive attribute of each source file after it is copied. If XCOPY can't fit all the desired files onto the floppy disk, you can simply place a fresh floppy disk in drive B and enter the command again:

```
XCOPY C:*.* B: /M
```

This time, XCOPY will ignore the files it copied previously, since DOS has turned off their archive attributes. ■

James Langdell is a technical writer for Sun Microsystems and author of the PC Resource MS-DOS Insert Collection.

A Farewell To Incompatibility

These four easy-to-use DOS commands help your hardware and software work together in peace.

by John Wolfskill

The problem of hardware and software incompatibility is as old as the personal computer is young. And chances are good it will remain an issue for a long, long time—most likely for as long as we own these complex machines. Such news provides little solace when you need to complete a task and your system just won't cooperate.

Consider these annoyingly common examples: Your reliable old spreadsheet program locks up and refuses to run under your new copy of MS-DOS 5.0. Another application, which arrived on a 5.25-inch disk, must be installed using drive A. But your drive A is designed for 3.5-inch disks. Worse yet, you bought a new floppy drive, only to find that your computer's ROM BIOS won't support its operation.

It may surprise you to learn that the solutions to these dilemmas are as close as your DOS directory. Four oft-overlooked utility programs—APPEND.EXE, ASSIGN.EXE, DRIVER.SYS, and SETVER.EXE—can bring peace to warring software and hardware components and greater tranquility to your computing sessions.

You Can Get There From Here

Many DOS users are familiar with the PATH command, a handy time-saver that tells DOS where to look for executable files (external commands, programs, and batch files) that aren't located in the current directory. Fewer DOS users, however, take advantage of PATH's companion utility, APPEND.EXE, which provides a way to specify a search route, or path, for any file. This, of course, includes the data files used by executable programs such as word processors and spreadsheets.

Let's say you want to find a data file located in a subdirectory. Ordinarily, to access that file you must switch to the directory where the file resides or type a long DOS pathname. APPEND lets you avoid this tiresome chore by tricking your applications into "thinking" that their data files are located in the current directory.

The first time you run APPEND, it loads into memory. Of course, unless you mention APPEND's subdirectory as an item in your PATH command, you'll need to include a full DOS pathname, such as the following, when you invoke APPEND:

```
C:\DOS\APPEND
```

Thereafter, APPEND becomes part of the DOS system kernel, remaining in memory until you shut off your system or remove the program from memory by typing:

```
APPEND ;
```

Append: Acts like the PATH command, providing DOS with a list of subdirectories to search if it can't find a data file in the current directory.

Command Type: External.

Available: DOS. 3.3 and later. Enhanced in DOS 4.0 to enable users to search for executable files as well as data files.

If you plan to use APPEND frequently, you'll find it convenient to place the following line in your AUTOEXEC.BAT file:

```
C:\DOS\APPEND C:\DOS
```

This makes APPEND and the files in your DOS directory available from any directory whenever you need to use them.

APPEND is useful in a variety of

situations that involve manipulating and accessing data files. After you've established an APPEND search list, you can access any file located in any subdirectory mentioned in the list by typing its name at the DOS prompt. For example, suppose your word processing data files are located in a subdirectory named C:\PW. To access a data file called MYFILE.TXT directly from the root directory of your hard drive, you can add the file's directory name to your APPEND list:

```
APPEND C:\PW
```

Thereafter, when you type:

```
TYPE MYFILE.TXT
```

from any directory, the file will be displayed on screen. To see the contents of the APPEND list at any time, you may simply type:

```
APPEND
```

and the list will be displayed. If no directories are specified in the current APPEND list, DOS will issue a "No Append" message.

You may think that an obvious use of APPEND would be to use the DIR command to locate and list the names of files located in other subdirectories. Unfortunately, DOS does not add filenames from appended directories to the directory listings that the DIR command produces.

As with the PATH command, you can specify a list of search directories as part of your APPEND list. Each item must be separated by a semicolon, as follows:

```
APPEND C:\;C:\PW;C:\BC7;C:\NORTON
```

Notice that I've specified the drive letter in each of the APPEND list entries. Although adding the drive letter to the search string increases the APPEND list's overall length, it lets you access the entries from any drive in your system. If you don't include a drive letter with each entry, DOS can access only those entries that reside on the current drive.

In addition, APPEND is useful for modernizing older DOS applications—typically, those written earlier than DOS 2.0—that don't recognize pathnames. Although these "old-timers" may be useful applications still, they force you to locate their data files in the subdirectory of the parent program. Using the APPEND command, you can move the data files into discrete subdirectories, then add the names of those subdirectories to your APPEND list.

Although APPEND provides a convenient way to access any file from any subdirectory, it has limitations. It can make your application read data files located in far-flung subdirectories, but it will not update those files. When the time comes to write the data file to disk, APPEND steps aside and lets your program write the updated file to the current directory. If this happens, you'll have two data files with the same name in two different directories. Worse, only one of the files will contain the updated information.

If you use DOS 3.3, you should remove APPEND from memory before you back up your hard drive with a disk backup utility. During the backup process, this version of APPEND may, upon finding an empty subdirectory, begin to search the APPEND list instead of taking the empty subdirectory at face value. The result can be a lot of disk space wasted backing up hundreds of files several times.

Read My Disks

While APPEND saves you time by making it easier to search for data files through myriad subdirectories, the ASSIGN command can save your sanity. It comes in handy when you're trying to install a DOS application that makes improper assumptions about the type of floppy disk drives you have in your system. As I mentioned earlier, the installation program of an application supplied on a 5.25-inch disk may ask you to insert the program disk into floppy drive A. If you've assigned a 3.5-inch drive as drive A, you've got a problem. Having a 5.25-inch drive as drive B won't help—unless you enlist ASSIGN.

ASSIGN neatly resolves this difficulty by convincing the application (and DOS) that your floppy drive B actually is floppy drive A. To swap the names of your drives temporarily, type:

```
ASSIGN B=A
```

before you begin installing the offending application. After you've installed the application, you can cancel the drive-letter reassignment by typing:

```
ASSIGN B=B
```

Of course, you may find other uses for the ASSIGN command, such as forcing an older DOS application that insists on writing its data files to a floppy disk mounted in drive A, to write those files to your hard disk instead. In this case the ASSIGN command is:

```
ASSIGN A=C
```

Be aware, however, that you can cause serious problems if an ASSIGN command is in effect during an operation that involves the DOS BACKUP, JOIN, LABEL, RESTORE, or SUBST command to retrieve file information. Likewise, you should never use the ASSIGN command while attempting to format or copy a disk with the FORMAT or DISKCOPY command. These DOS commands ignore drive letter reassignments completely.

ASSIGN: Redirects input/output (I/O) operations from one disk drive to another.

Command Type: External.

Available: DOS 2.0 and later.

Finally, never use ASSIGN to redirect the output of your hard disk drive to another drive. The results could be disastrous.

In the Driver's Seat

DRIVER.SYS is a seldom-used DOS utility that can be helpful in a variety of situations. And you'll enjoy its ability to convince DOS that a single floppy disk drive actually is two different disk drives.

DRIVER.SYS: Creates a logical drive that you can use to refer to a physical floppy drive.

Program Type: External.

Available: DOS 3.2 and later.

For example, assume yours is a typical IBM-compatible system equipped with a 1.2MB 5.25-inch floppy drive assigned as drive A, a 720K 3.5-inch drive assigned as drive B, and a hard drive assigned as drive C. If you want to copy to another floppy disk a group of files on a 1.2MB disk, things get sticky if the files occupy more than 720K of disk space.

You can copy the files to your hard drive, insert another 1.2MB disk in drive A, copy the files from the hard drive to the new disk in drive A, and erase the copied files from your hard drive. Or you can use DRIVER.SYS to create two new logical drives—drives D and E—so that you can use drive A to copy files directly from one 1.2MB disk (drive D) to another (drive E). This technique works with both the COPY and XCOPY commands.

To lay the groundwork, you must install DRIVER.SYS in your CONFIG.SYS file, inserting one DEVICE statement for each logical floppy disk drive you want to create. In the current example, you must install DRIVER.SYS twice, because copying files requires a reference to two logical drives. The syntax to do this is as follows:

```
DEVICE=C:\DOS\DRIVER.SYS /D:0 /F:1
DEVICE=C:\DOS\DRIVER.SYS /D:0 /F:1
```

DRIVER.SYS requires that your DEVICE statement include two command-line switches to tell DOS how to configure the logical drives. The /D switch refers to the number of your physical disk drive. Floppy drive A is always referred to as drive 0, floppy drive B is drive 1, and an external drive is always assigned the number 2. The /F switch (the *f* stands for *form factor*) tells DOS the capacity of the physical drive (see "Form Factor Equivalents").

In the above example, note that both DEVICE statements are the same. That's because you're defining floppy drive A to serve as both logical drives D and E. DOS assigns the logical drive letters automatically each time you restart your system, using the letters of the first available logical drives. Since physical drives A, B, and C already exist in the system described above, the next two available logical drive letters are D and E.

After you add the new DEVICE statements to your CONFIG.SYS file and reboot the system, you're ready to transfer the contents of a 1.2MB floppy disk mounted in drive A to another 1.2MB disk mounted in the same drive. Start by inserting the 1.2MB disk containing the files to be copied into drive A (the source disk) and logging onto drive D by typing:

```
D:
```

At the D:> prompt, type:

```
COPY D:\*.TXT E:
```

When DOS displays an "Insert a disk for drive D: then press the <enter> key" message, simply press Enter, because the correct disk already is in the drive. The drive's light will go on and then off; then DOS will tell you to insert the disk for drive E and press Enter. Remove the first disk from drive A, insert the disk to which you want to copy files (the target disk), then press Enter.

DOS will continue to ask you to swap disks in drive A until all of the files are copied. If more than a few files are involved, the process is time-consuming and it may be more efficient to copy the files to your hard disk and then back to a disk in drive A. But if you don't have a hard drive, this technique is probably the best way to get things done without purchasing a special utility program.

If you use DOS 4.0 or later, DRIVER.SYS and its companion DOS command, DRIVEPARM, can help out when you add a drive that your computer's built-in ROM BIOS does not support. Let's say you install a new 1.44MB 3.5-inch floppy drive in a system that contains a single floppy drive and a hard disk drive, only to discover that the new drive doesn't work after you restart your computer. To solve the problem, simply add the following lines to your CONFIG.SYS file:

FORM FACTOR EQUIVALENTS

Number	Disk Capacity	Disk Size
0	360K	5.25-inch
1	1.2MB	5.25-inch
2	720K	3.5-inch
7	1.44MB	3.5-inch
9*	2.88MB	3.5-inch

*DOS 5.0 users only.

```
DEVICE=C:\DOS\DRIVER.SYS /D:1 /F:7
DRIVEPARM=/D:1 /F:7 /I
```

The first line tells DOS your new drive is attached to your floppy disk controller as the second drive in the chain (physical drive B), and that you're installing a 1.44MB drive. After DRIVER.SYS creates a logical drive letter for the new drive (in this case the letter D), DRIVEPARM's /I switch tells DOS your new drive is an electronically compatible 3.5-inch drive that isn't supported by your computer's built-in ROM BIOS.

After you reboot your system and allow the new commands to take effect, your new 1.44MB floppy drive will spring to life as floppy drive D.

Get Your Apps in Line

SETVER, a new command added to DOS in Version 5.0, provides a way to coax an application program written to work specifically with an older DOS version into working with DOS 5.0.

For compatibility reasons, many applications check for the installation of a particular DOS version before they attempt to start. If DOS passes an incorrect version number to a program, the program may balk and refuse to run. Although there may be no reason (other than the incorrect version number) for the program not to work, you'll be stuck with an application you can't use.

SETVER neatly solves this problem by keeping a list of your programs in a special file called the version table. Each entry in the table contains the application's name and the DOS version for which it was designed. If you encounter a program that refuses to start, try adding its name and required DOS version to the version table. The next time the program requests the DOS version number, DOS will pass the required number instead of its own.

TIP A TIMELY PROMPT

If you would like DOS to help you keep track of time, try adding the following line to your AUTOEXEC.BAT file:

```
PROMPT $T$H$H$H$H$H$H $P$G
```

The $T parameter tells DOS to read the current time from your computer's internal clock; the time is then displayed in the system prompt. The six $H commands—the PROMPT parameter for backspace—instruct DOS not to include seconds and tenths of a second in your on-screen clock.

—J.N.

SETVER: Assigns the MS-DOS version number that DOS reports to an application program.

Command Type: External.

Available: DOS 5.0.

In many cases, this software trick convinces the program that it's running under an older version of the operating system and the application magically springs to life. I've found SETVER very useful for reviving application programs that were designed to operate under older versions of Microsoft Windows and initially refused to work under Windows 3.0.

To use SETVER effectively, you must know the specific DOS version for which your application is designed. Sometimes this is not easy to determine. If you can't find this information in the manual supplied with your application, a bit of detective work may be in order. Start by trying to determine the year in which you purchased the application, then compare that date with the DOS version current at that time (see the table "Version Control" at right). For example, an application you purchased in 1987 most likely would fall within the realm of MS-DOS 3.3 or earlier. Keep in mind, however, that the dates shown in the table are approximate. You may need to experiment with several entries until you satisfy the application's appetite for a particular DOS version.

Before you can use SETVER, you must load the version table into memory by adding the following line to your CONFIG.SYS file:

```
DEVICE=C:\DOS\SETVER.EXE
```

DOS 5.0's setup program automatically adds this line to your CONFIG.SYS file. (You can, however, opt not to have your CONFIG.SYS file altered during installation. You may want to check to make sure it exists, however, before you begin. If the entry doesn't exist, add it, then reboot your system.

VERSION CONTROL

MS-/PC-DOS Version	Release Date
1.0/1.25	08/81
2.0	03/83
2.1/2.11	11/83
3.0	08/84
3.1	03/85
3.2	01/86
3.3	04/87
4.0/4.01	07/88
5.0	09/91

After the version table is loaded into memory successfully, you can list the entries it contains by typing:

```
SETVER D:
```

where D: is the letter of the current drive. Type the following line to add the name of your application and its DOS version to the table:

```
SETVER appname n.nn
```

where *appname* is the name of the application and *n.nn* is the DOS version for which it was designed. If, for example, a program were designed to work under DOS 3.3, you would substitute 3.30 for *n.nn*. After adding the entry to the table, you must reboot your system to make the changes effective. Then stand back, cross your fingers, and try to start the offending application. ■

John Wolfskill is a free-lance writer and president of Powertrain Development, a New Hampshire-based software development company.

10 Worst DOS Commands

Although Microsoft Corp. has introduced a wealth of worthy DOS commands in its many releases, the company has thrown a few bones into the bag, as well.

by Dan Gookin

Over the years, computer journalists have devoted articles, magazines, and entire books to extolling the wonders of DOS—and rightly so. DOS has put powerful tools at our fingertips and has helped change computing forever. Just the same, a number of DOS commands are inherently confusing, downright ill-behaved, or potentially lethal to your system.

Consider the SWITCHES configuration command. It does two separate and unrelated things, and neither of them has anything to do with the word *switches*. How's that for confusion? SETVER, introduced with DOS 5.0, encourages developers not to produce and distribute newer versions of their software. The command "fools" older, incompatible software into running on the new DOS version, but it also rings in the potential for disaster when used inappropriately: Forcing an older DOS program to run using SETVER can crash your system.

But these commands earn mere dishonorable mentions in the DOS Hall of Shame. The following 10 commands—listed from bad to worst—have earned their place because they beg to be ignored, overhauled, or banished to bit hell.

My purpose in singling out misfits is not to ridicule DOS or Microsoft. DOS has shown a lot of improvement over the past few versions. Older, cruddy commands stick around for compatibility reasons (witness Edlin). But, as long as there's hope, there's also room for criticism—and warning about potentially deadly DOS commands.

#10 EDLIN

Edlin was DOS's original text editor. In fact, until it was replaced with DOS 5.0's Edit full-screen editor, it remained virtually untouched since its first appearance with PC-DOS 1.0 in 1981. Edlin is awkward and ugly. The only thing nice about it is that it's the only text editor common to every single DOS computer.

EDLIN: DOS's line-oriented text editor.
Command Type: External.
Available: DOS. 1.0 and later.

Because picking on Edlin is in vogue, and because DOS 5.0 now comes with EDIT.COM, I have nothing more to say on the subject.

#9 Code-Page Commands: CHCP, NLSFUNC, MODE, and others

I have nothing against the code-page commands. If you need access to foreign-language character sets on your

CHCP: Displays the number of active code pages or changes the active code page used with your current setting.
Command Type: Internal.
Available: DOS 3.3 and later.

NLSFUNC: Loads country-specific information for national language support.
Command Type: External.
Available: DOS 3.3 and later.

MODE: Gives you control over certain aspects of setting up your printer, display, and serial ports.
Command Type: External.
Available: DOS 3.2 and later.

PC; then these commands are miraculous; they load in the special "code pages"—graphics symbols and letters used in other languages—and let you use the keyboard to access them easily. This is a versatile and handy aspect of DOS. Its implementation, however, stinks.

TIP MEMORY MAGIC

DOSKEY, the DOS 5.0 command-recall utility, normally sets aside 512K, or .5MB, of temporary memory for its command buffer. This lets you save approximately 35 commands that have an average length of 35 characters. You can use the /BUFSIZE parameter to increase or decrease the size of your DOSKEY buffer. For example, the following command:

```
DOSKEY /BUFSIZE=256
```

decreases the size of the DOSKEY buffer to 256K, thus freeing memory for other uses.

—J.N.

What DOS really needs is one, handy code-page command. Instead, it offers four commands: CHCP, NLSFUNC, variations on the MODE command, and KEYB. This doesn't include the COUNTRY command in CONFIG.SYS, or the COUNTRY.SYS and KEYBOARD.SYS device drivers, or the tables and tables of code-page numbers and country codes. Sheesh. Isn't a computer supposed to make things easier?

#8 FC

You may be unfamiliar with this command. I was, since I use PC-DOS instead of MS-DOS.

FC: Provides a limited but more capable file-comparison utility than the COMP.CON program that comes with PC-DOS.
Command Type: External.
Available: MS-DOS 2.0 and later.

Essentially, FC is a more powerful file-comparison program than COMP.COM. You can almost choke on this command's mile-long parade of options and switches. Why is all that necessary just to compare two files?

Although FC works, my primary grudge is its inelegance. I prefer a file-comparison utility I found on a BBS a while back. It graphically shows which portions of two files are different. That's all I want to know. I don't need to waste my time setting FC's options.

#7 CTTY

The CTTY command looks good on paper, and it has inspired a lot of DOS novices with dreams of remote computing or hooking up a second "terminal" to their PCs. Yet, CTTY appears to be a vain attempt at emulating a more cryptic Unix command that configures the terminal. (*TTY* stands for *teletype*, which, on a DOS computer, is your keyboard and screen.)

CTTY: Changes the terminal device used to control your system
Command Type: Internal.
Available: DOS 2.0 and later.

When you use DOS, it accepts input from the keyboard and displays output on the screen. Those two devices are collectively referred to as the console, which DOS calls the CON device. Using the CTTY command, you can specify a different device for DOS's input and output. Of course, since the device must be capable of interactive input and output, this rules out the printer and disk drives, which leaves you with the serial port. When they uncover this, most people stop reading and start dreaming, "Gee, I can hook up a second monitor and keyboard to my computer..." But wait!

True, you can hook up a second monitor and keyboard, such as a laptop computer (and I've done it with a Macintosh), using a null-modem cable to connect the two machines' serial ports. You then configure the serial ports using the MODE command under DOS and run a communications program on the second computer. Typing the command:

```
CTTY AUX
```

tells DOS to use the serial port COM1 for input and output. And, lo, on the second computer you'll see the DOS prompt and be able to use DOS commands.

The drawback is that you can use only DOS commands on the second computer, and only text commands at that. If you run a program, it will appear on the first PC. So, first off, CTTY is not a remote-control program for your computer. Second, you cannot transfer files between the two computers, or do much of anything for that matter. (Sure, text-file transfer is possible, but other programs can do this better (see the sidebar, "Winning Alternatives" on page 26.)

The deadly side of CTTY surfaces when you don't have anything hooked up to your serial port and you type CTTY AUX anyway. This permanently disables your keyboard. Since you need to type CTTY CON to regain control but can't, you must reset instead. Let's face it, CTTY is just a useless command, fun to demonstrate but impractical.

#6 MODE

The MODE command is highly useful. It's too useful, in fact. MODE seems to have been designed by someone at the Pentagon. It does 12 different things, each time using a different format. A soldier in the field would be shot before he could assemble and use the MODE command. DOS users would rather inhale bus fumes than read the myriad MODE-command instructions.

MODE: Gives you control over certain aspects of setting up your printer, display, and serial ports.

Command Type: External.

Available: DOS 3.2 and later.

What would have been better is 12 individual utilities, each designed to handle a specific task, as opposed to the general-purpose MODE command. For example: A single command to set the code pages; a command to set up the serial ports; a printer setup command; a command for the screen; and, well, you get the idea.

#5 DISKCOMP

I've hated this DOS command since the days of the floppy-only PC. There's only one reason for the DISKCOMP command: to compare two floppy disks; it doesn't work with any other type of disk.

Typically, you'd want to use DISKCOMP only after using the DISKCOPY command to duplicate a disk. So, is Microsoft telling us that DISKCOPY is that unreliable?

Some may claim that DISKCOMP verifies that DISKCOPY didn't encounter any errors. Wrong! DISKCOPY tells you that right away. If the disk is defective, the copy stops abruptly. Okay, so then you use DISKCOMP to compare two disks after they've been copied, say two months down the road. Now why would anyone want to do that? It's just easier to use DISKCOPY again, since, if DISKCOMP fails, that's what you need to do anyway.

DISKCOMP: Compares two floppy disks byte for byte to ensure they have identical contents.

Command Type: External.

Available: DOS 1.0 and later.

Even if you can dream up another plausible reason for DISKCOMP, say to check for errors (or something equally stretched), the process just takes too long to complete. DISKCOMP involves a lot of elbow work and disk swapping. In the age of the hard drive, little justification exists for this command.

#4 EXPAND

Introduced with DOS 5.0, EXPAND is designed to decompress files shipped on the DOS distribution disks. This is very trendy, since most other major applications come in compacted form, as well. What's dumb are the EXPAND program's non intuitive format and its lack of a companion CONTRACT or COMPRESS program. This makes EXPAND useless after you've installed DOS.

EXPAND: Provides a means of decompressing files on DOS 5.0's installation and update disks.

Command Type: External.

Available: DOS 5.0.

Let me get picky here. To expand a program under DOS, you must specify the name of the compressed file plus the name of the file after it has expanded. Third-party compression programs have always stored the original filename in the compressed file, eliminating the need to specify both. Why then is EXPAND so klutzy?

And why did Microsoft bother creating such a crippled program? Data-compression and file-archiving software, which is what EXPAND really should be, are great utilities to have—modem users couldn't live without them. Microsoft could easily have bundled both the compression and decompression program with DOS.

#3 APPEND

I've read three software manuals already that scream, "Don't use the APPEND command!" Basically, APPEND sounds like a good idea: Just as the PATH command lets you look for programs in other directories,

WINNING ALTERNATIVES

DOS is inspiration to thousands of programmers. They look at it and wistfully sigh, "I can do that better." For example, if you see potential in the CTTY command, then you should look into the several PC-based remote communications programs. PC Tools has one called Remote, and there's also the popular PC/ Anywhere from Symantec, as well as Microcom's Carbon Copy Plus. For plain file transfer between two computers, you have your choice of Traveling Software's LapLink Pro, Fifth Generation's Brooklyn Bridge, PC Tools' DeskConnect, and The Norton Commander's Link option. You can find all of these at your local software store.

Though Edit has replaced Edlin as DOS's text editor, I'm a big fan of QEdit. It takes less disk space than Edit, lets you work with multiple files, and QEdit is ideal for laptops.

If you need a handy tool for compressing—and decompressing—files, then the ultimate programs are PKWARE's PKZIP and PKUNZIP utilities. You can find this package on national on-line systems, if you have a modem. Most local computer clubs keep a copy in their disk libraries.

The COMP utility I use instead of DOS's COMP command, or (ugh!) FC, is Michael J. Mefford's COMPARE program, written for *PC Magazine*. You can get it from the PC MagNet forum on CompuServe or your local computer club library.

DOS's SMARTDRV command can replace FASTOPEN, but Multisoft's PC Kwik cache, The Norton Utilities' NCache, and Golden Bow's Vcache are all faster and better. Instead of using RECOVER, try any of the popular disk utilities, including The Norton Utilities, PC Tools, Steve Gibson's SpinRite, and Disk Technician Advanced from Disk Technician Corp.. You also can find all of these at your local software-o-rama.

Unfortunately, nothing that I know of replaces the MODE command.

—D.G.

PRODUCT INFORMATION

Brooklyn Bridge,
$139.95
Fifth Generation
10049 N. Reiger Rd.
Baton Rouge, LA 70809
800-873-4384
504-291-7221

Carbon Copy Plus,
$199
Microcom
500 River Ridge Dr.
Norwood, MA 02062
800-822-8224
617-551-1000

Disk Technician Gold,
$149.95
Disk Technicians Corp.
1940 Garnet Ave.
San Diego, CA 92109
800-847-5000
619-274-5000

LapLink Pro,
$169.95
Traveling Software
18702 N. Creek Pkwy.
Bothell, WA 98011
800-343-8080
206-483-8088

PC Anywhere,
Host-remote $179;
LAN $495
The Norton Commander,
$149
The Norton Utilities,
$179
Symantec Corp.
2500 Broadway
Suite 200
Santa Monica, CA 90404
800-441-7234
310-453-4600

Super PC Kwik,
$79.95
PC Kwik Power Pak
$129.95
Multisoft
15100 S.W. Koll Pkwy.
Suite 200
Beaverton, OR 97006
503-690-8090

PC Tools,
$179
Central Point Software Inc.
15220 N.W. Greenbrier Pkwy.
Suite 200
Beaverton, OR 97006
800-445-4208
503-690-8090

PKZip, PKUNZIP and PKSFX Package,
$47
PKWare
9025 N. Deerwood Dr.
Brown Deer, WI 53223
414-354-8699

QEdit advanced,
$54.95
Semware
4343 Shallowford Rd.
Marietta, GA 30062
404-641-9002

SpinRite,
$89
Gibson Research
35 Journey
Aliso Viejo, CA 92656
714-362-8800

VCache,
$69.95
Golden Bow Systems
P.O. Box 3039
San Diego, CA 92163
800-284-3269
619-298-9349

APPEND lets you look for data files. Neat-o. But this command has a dark side that makes it easily misused, even dangerous.

APPEND: Provides a list of subdirectories for DOS to search if it can't find a data file in the current directory.

Command Type: External.

Available: DOS 3.2 and later.

With APPEND loaded, your programs and DOS commands can locate data files in other directories. So, if you want to start WordPerfect and load the MEMO2-18.WP file, you simply type:

```
WP MEMO2-18.WP
```

It doesn't matter if MEMO2-18.WP is not in the current directory; with APPEND loaded, DOS will search the directories on the appended path to locate that file. This is great, especially for people who don't understand subdirectories.

Now consider another scenario: You want to delete all the .DOC files in the current directory. You type `DEL .DOC`, but it takes a bit longer to run than you anticipated. Why? Because APPEND has diligently searched out all the .DOC files on the appended path—and efficiently deleted each of them. Ugh!

While it's true that you can prevent the APPEND command from doing that, it's also true that APPEND's many switches, as well as its general purpose in life, require a lot of explanation. My advice is to use your subdirectory system and delete APPEND.EXE from your DOS directory.

#2 RECOVER

Another DOS dog, RECOVER, has a miraculous-sounding name, but it's a notoriously nasty utility. Basically, RECOVER is designed to rebuild a file that happens to lay over a bad sector of the disk. It does this with ease but with little grace: The file is renamed FILE*xxxx*.REC, and the missing or corrupted section is replaced with empty bytes or zeros. Using a gross simile, if a file were a snake, RECOVER would sew in a sausage for the missing piece and still try to convince you it was all snake.

The worst example of RECOVER's deadliness is when you use it on a hard drive. RECOVER sees these things called subdirectories and assumes they're all bad files. So it converts each of your subdirectories into a FILE*xxxx*.REC file. Ta-da! Instantly every file on your hard drive is effectively gone. Why does Microsoft keep such a deadly program around?

RECOVER: Offers a last-ditch recovery method for salvaging readable parts of a file stored in damaged disk sectors or reclaiming undamaged sections of a disk with a damaged directory.

Command Type: External.

Available: DOS 2.0 and later.

To prove that RECOVER.EXE should be deleted from your hard drive, I should point out that the many disk-repair utilities in both Central Point Software's PC Tools and Symantec's The Norton Utilities are written specifically to recover from the RECOVER command.

#1 FASTOPEN

Here's one you can hurl into the abyss. FASTOPEN is perhaps the worst DOS command ever written. It's number one on my bottom-10 list—even more so since some DOS installations automatically stick FASTOPEN into your CONFIG.SYS or AUTOEXEC.BAT file. That hurts.

FASTOPEN is sinister. It can cause more damage than good, especially when used with disk utilities such as defragmenters, optimizers, and compression utilities like Stacker...the list is endless. The manual for The Norton Utilities begs you to delete FASTOPEN.EXE from your hard drive "at once!"

FASTOPEN—Stores hard drive directory information so that DOS can open files faster.

Command Type: External.

Available: DOS 3.3 and later.

FASTOPEN saves a list of files you've recently opened on your computer. It remembers where the files are located on disk. So, if you need to access those files a second time, they're opened a wee bit quicker. The benefits here aren't that great, but the potential for disaster is.

Here's a FASTOPEN jewel for you: Say you've decided to use FASTOPEN on your sluggishly slow floppy drives. After a time, you notice that those files in drive A seem to fly off the disk. Then you commit the FASTOPEN sin of changing disks. Oops! Now you'll see an error message that warns you of an unauthorized disk change: "Please insert volume xxxxx serial xxxx-xxxx." You'll see that message over and over until you stick the original disk back into the drive—or reset. This is not a boon to productivity. Enough said!

Dan Gookin is the author of DOS for Dummies (IDG Worldwide), The Microsoft Guide to Managing Memory with DOS 5 (Microsoft Press), *and the* DOS 5 User's Guide (M&T Press).

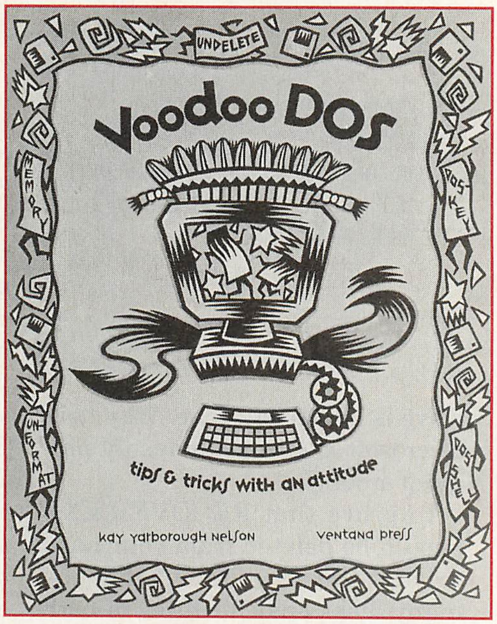
DOSKEY Revealed

Need to repeat a series of DOS commands? DOS 5.0's macro generator lets you do that — and more — with ease.

by Kay Yarborough Nelson

If you like playing with batch files, you'll love DOS 5.0's DOSKEY feature. It's a terminate-and-stay resident (TSR) program that remains in memory after it has started to let you create macros. Macros—simply batch files by another name, except that they're much faster—get lost when the lights go out or when you turn off your computer. Later in this article you'll find a way around that.

You can do more than create macros with DOSKEY, however. (In fact, a lot of folks—those who don't like batch files—never even want to get into DOSKEY macros.) There's an even simpler way to put DOSKEY to use: Because it remembers the commands you type, you can use them again without having to enter them again. You may prize DOSKEY most for this. In addition, DOSKEY lets you issue several commands on a single line instead of demanding that you follow the standard "one line, one command" rule. Now that's magic—at least, for DOS.



DOSKEY: Maintains a "history" of commands issued from the C:> prompt through which you can scroll. Lets you type and run several DOS commands from one command-line instruction. In addition to creating these "compound" commands, DOSKEY lets you build recallable macros.

Command Type: External, but after it is loaded, it acts like an internal command.

Available: DOS. 5.0.

You can use DOSKEY in a number of distinct ways:

- To give several commands at once, all on the same line
- To repeat previously issued commands
- To edit previous commands and use them again in a slightly different form
- To create macros in DOS
- To create batch files from your macros

This article initiates you into voodoo DOSKEY by taking a look at the uses above and giving you tips, and in some instances traps, associated with each of them.

Giving Several Commands At the Same Time

It's certainly faster to type everything you want DOS to do and press Enter than to type a command, press Enter, wait, type the next command, and so on. If you're comfortable with the command se-

Editor's Note: *The following is excerpted from Kay Yarborough Nelson's book,* Voodoo DOS: Tips & Tricks with an Attitude, *recently released by Ventana Press. Reprinted with permission of Ventana Press. All rights reserved.*

quence you're entering, why not use DOSKEY and enter everything at once?

TIP LOADING DOSKEY.

DOSKEY normally isn't present when you start your computer. To put it in memory, type:

```
DOSKEY
```

at the command line.

If you're not always working through the DOS shell (see the trap below), why not put the DOSKEY command in your AUTOEXEC.BAT file? That way, a history of the commands you execute in any one session is always available. It's a lot easier to press the up-arrow key to recall your previous command (and edit it, if you like) than to type it over again. DOSKEY takes up 4K of RAM, though, so don't put it in your AUTOEXEC.BAT file if memory is at a premium.

TRAP DON'T TRY DOSKEY IN THE DOS SHELL.

DOSKEY works only at the command line, not in DOSSHELL. If you don't put it in your AUTOEXEC.BAT file, load it from the command line.

TIP ISSUING SEVERAL COMMANDS ON A SINGLE LINE.

After DOSKEY is loaded, you can type several commands at once, separating each with a Ctrl-T. (Ctrl-T puts a paragraph mark (¶) on your screen.) Press Enter to end all the commands.

For example, to move a directory from one place to another from the command line, you have to copy all the files in it to another directory, erase the files in the original directory, and delete the original directory. (Remember, though, that DOSSHELL lets you rename directories.) To move a directory called C:\FILES to C:\REPORTS\SEPT using DOSKEY, simply follow these steps:

- Load DOSKEY
- Type `COPY *.* C:\REPORTS\SEPT` while you're in the FILES directory, press Enter, then press Ctrl-T
- Type `DEL *.*` , press Enter and press Ctrl-T
- Type `CD..`, press Enter, and press Ctrl-T
- Type `RD FILES` and press Enter

You'll be asked to confirm that you really want to delete all the files. When you do, DOS will go ahead and move the directory (copy, delete, and remove it).

TIP CORRECTING A MISTAKE.

If you're typing a long sequence of commands and you realize, before you press Enter, that you made a mistake, you usually backspace over the offending characters and retype the line. But with DOSKEY you can use the arrow keys to move to the mistake and correct it, which is a lot faster than backspacing and retyping.

If you've already used the command and you're recalling it from DOSKEY's history buffer (see below), there are several ways to edit it.

Recalling Previous Commands

Even if you don't want to use DOSKEY to put more than one command on a line, you probably will want to use it to recall commands you issued previously. This feature can really save time, because a lot of your work at the DOS prompt—such as copying bunches of files onto floppy disks or getting a listing of different directories—consists of doing pretty much the same thing again and again.

After you load DOSKEY, it automatically keeps track of each command you issue. You can recall any of these commands and use them again, or recall and edit them for use with your current task.

TIP PRESSING THE UP-ARROW KEY TO REPEAT A COMMAND.

When DOSKEY is installed, pressing the up-arrow key repeats the last command you gave. Pressing the up-arrow key again gives you the command before that, and so on.

In DOS, F3 brings back the previous command, but there is a difference with DOSKEY: It keeps a history of all the commands you've used, to a certain point (limited by the buffer size). If you press PgUp, DOSKEY displays the oldest command in the history list; pressing PgDn displays the latest command.

TIP ERASING THE CURRENT COMMAND.

To get rid of the command currently displayed, press Esc, or PgDn and the down-arrow key.

TIP VIEWING THE HISTORY LIST.

To see the whole history list, press F7. You then can choose which command you want to use again from this list, either by highlighting it with the up- and down-arrow keys and pressing Enter, or by pressing F9, typing the number of the command, and pressing Enter twice.

If you entered the example described in a previous tip for issuing several commands on a single line, you would press F7 to see the following commands, numbered in the order you entered them:

```
1. COPY *.* C:\REPORTS\SEPT
2. DEL *.*
3. CD..
4. RD FILES
```

You actually may see different commands, depending on what you've done since loading DOSKEY.

TIP SPEED SELECTING.

There's an even faster way to use a command from the list DOSKEY displays: Press Esc, type the first few letters of the command, and then press F8. You can cycle through the commands that begin with the same letters by pressing F8 again.

TIP RECALLING THE NEXT COMMAND IN THE LIST.

Assuming you've moved back somewhere in the DOSKEY list to use a command issued previously, you can press the down-arrow key to display the next command in the list.

TIP CLEARING THE HISTORY LIST.

To clear the buffer of all previous commands and start a new list, press Alt-F7.

TIP CHANGING THE NUMBER OF COMMANDS DOSKEY HOLDS.

The default buffer size is 512 bytes. If you think of a byte as a character, that's already a lot of commands. To change the buffer size so it will remember even more commands, start DOSKEY with the /BUFSIZE= switch and specify a new size greater than 512.

Editing DOSKEY Commands

You probably don't want to issue the same command over and over. DOSKEY has all sorts of rather arcane editing tricks to modify previous commands for future use. Because these shortcuts are easier to use than to read about, you might want to start DOSKEY, issue a few commands, and practice along with these tips.

TIP EDITING A COMMAND TO USE IT AGAIN.

Once you've displayed a DOSKEY command, you can edit it. You might want to specify a different filename or wildcard pattern, or you might want to move files to a different directory. Use the arrow keys to move to a character and use the Backspace key or the Del key to delete characters. For example, it's easier to change `COPY .TXT A:` to `COPY *.DOC B:` by changing the .TXT and the A: than it is to retype the whole command.

TIP SPEED EDITING.

There are a lot of neat shortcuts for editing a DOSKEY command. For instance, if you press F2 and type a character, the previous command—up to the character you just typed in—is inserted. When you press F3, the rest of the previous command is inserted. If the previous command were:

```
DIR *.BAT /S /O:D
```

you could press F2 and type a period (.) to get DIR *. Then type:

```
COM
```

to change BAT to COM. Press F3 to get the rest of the previous command.

Here's the neatest trick of all, in my opinion: Simply press the right-arrow key (or F1) to insert the characters from the previous command one at a time, or press Ctrl and the right-arrow key to recall one word at a time. The F2 and F3 tricks above require a little thought on your part, but this one doesn't! The previous command just appears, character by character. When you get to the part you want to change, press the Ins (insert) key and type the new characters. You then can recall more characters from the previous command with the right-arrow key or F1.

TIP EDITING WORD BY WORD.

If a command is full of switches (such as `XCOPY C:\REPORTS A: /S /M /P`), the quickest way to move between them is to press Ctrl-right arrow or Ctrl-left arrow. To DOS, anything separated by a space is a word.

TIP EDITING A COMMAND DOESN'T CHANGE COMMANDS IN THE BUFFER.

Editing a command changes only the new command you're entering—not the commands already in the buffer. So feel free to edit away.

TIP GETTING HELP ON DOSKEY.

Don't try to memorize everything about DOSKEY and its special symbols. You're already at the command line, and you can get a quick reminder of DOSKEY features by simply typing:

```
HELP DOSKEY
```

or:

```
DOSKEY /?
```

Using DOSKEY for Macros

DOSKEY also lets you define or create macros. A macro is simply a set of instructions you give to DOS. Defining a macro is just slightly different from using DOSKEY to issue several commands on one line. You type DOSKEY, the name you want your macro to have, an equal sign (=), and then the commands you want the macro to perform. But instead of separating commands with Ctrl-T (as you do to put several commands on one line), use a dollar sign—either an uppercase *$T* or a lowercase *$t* will work.

Suppose you want a directory command that lets you see your document files first (those with a .TXT extension) and then your spreadsheet files (those with a .WKS extension), sorted by date. Set up a macro called DOIT that sorts a directory listing by date, such as:

```
DOSKEY DOIT = DIR .TXT /O:D $T DIR *.WKS /O:D
```

TIP RUNNING YOUR MACROS.

To run a macro you've created, type its name at the command prompt and press Enter, just as you would with any DOS command.

TIP STOPPING A MACRO.

Ctrl-C or Ctrl-Break stops the current macro command from executing. You may have to press it more than once to cancel the whole macro.

TIP CUSTOMIZING DOS COMMANDS BY DEFINING THEM AS MACROS.

If you give a macro the same name as a DOS command (such as COPY or ERASE), DOS executes the macro instead of the command. DOSKEY first looks for and expands macros before passing the expanded command line to DOS.

Would you like to customize DOS commands to do what you want them to? Suppose you always want your directory listings sorted by date. Define a macro (name it DIR) as DIR /O:D, which sorts listing by date, and every time you issue the DIR command your directory listing will be sorted by date.

If there are DOS commands you always use with the same switches—such as `XCOPY /S /E B:` (which copies to drive B everything in the current directory, along with all the related subdirectories and even any empty subdirectories)—you can create a macro with the same name as the command for parallel construction—XCOPY, in this case. Then you won't have to remember the switches.

To retain the ability to use the original DOS command, even though you've set it up with a bunch of switches as a macro, just type a space after the command prompt and then type the command name. (This could also be a trap, depending on how you look at it.)

TIP USING MACROS AS SHORTHAND.

You can set up a macro as a shorthand notation for a longer command. Suppose you are like me and tend to type DOSSSHELL (one *s* too many). Type the following line:

```
DOSKEY SH = DOSSHELL
```

to abbreviate DOSSHELL to SH. Then you have only to type SH at the command line to enter the shell.

This is good for any command that's tedious to type or hard to remember. To make your shorthand notation permanent, see the tips about turning your macros into batch files, later in this article.

Macros vs. Batch Files

DOSKEY macros are similar to batch files, but there are some pretty important differences. For one thing, macros exist only in RAM, and because RAM is much faster than a disk drive, it carries out your commands faster than batch files. The most important difference, though, is that, as with everything in RAM, macros disappear when you power down.

Keep these considerations in mind when you choose between making a procedure into a macro or into a batch file.

TIP MACROS HAVE SIZE LIMITS.

You can use only 127 characters in any one macro. Still, that's a pretty big macro! Batch files, on the other hand, can be as long as you like.

TRAP YOU CAN'T USE THE GOTO COMMAND IN MACROS.

Because you are unable to use the GOTO command in macros, if you want to branch to different sets of commands, use batch files instead.

TIP MACROS USE DIFFERENT SYMBOLS AS REPLACEABLE PARAMETERS.

Batch files use the symbols %0 through %9 as replaceable parameters; macros use $0 through $9.

TIP MACROS CAN START BATCH FILES.

A macro can't start another macro, but it can start a batch file. A batch file, however, can load your saved macros into memory so that you can use them on more than one occasion.

TIP MACROS ARE FASTER THAN BATCH FILES.

Because macros exist only in RAM, they're faster than batch files, which are stored on disk. If you use a routine often (and it doesn't mind the limitations described above), make it a macro instead of a batch file. Read the next tip to learn how to save your macros.

Saving Macros

Macros exist only in RAM, so, as mentioned before, they disappear when the power fails or when you turn off your computer. You *can* save your macros, however, as you'll see in the following tips.

TIP TURNING MACROS INTO BATCH FILES.

If you turn a macro into a batch file, you can use it at any time. To do this, direct your macro to a text file and then edit it to make only the lines you want to use into individual batch files. For example, to store a bunch of macros you've just created in a file you want to call MACROS.DOC, you would type:

```
DOSKEY /H > MACROS.DOC
```

If you enter:

```
TYPE MACROS.DOC
```

to see what's in the file, you'll find that it contains all the commands you've used, including the commands that defined the macros. (Think of the /H switch as short for *history*.)

You then can use a text editor to convert the macro commands in MACROS.DOC into batch files.

First, edit the MACROS.DOC file to delete any lines you don't want, then save (as a batch file with a .BAT extension) each group of commands you do want (see the next tip).

TIP USING THE /H SWITCH TO CHANGE MACROS INTO BATCH FILES.

When you redirect macros to a file with the /H switch, each line already will begin with DOSKEY, so you won't need to change that to make the macros into a batch file. For example, a line might appear this way:

```
DOSKEY DOIT=DIR .TXT /O:D $T DIR *.WKS /O:D
```

Because this is a perfectly good DOS command, all you need to do is save it in a batch file.

But look closely to see if there's any more editing to be done. Each command must begin on a separate line, for example, and the commands must follow a syntax DOS can understand. There may be lines you don't want to include in your batch file, such as the DOSKEY /H line that sent the macros to the file in the first place.

Suppose you want to save the following macro, which displays your word processing documents and spreadsheets by date:

```
DOSKEY DOIT=DIR *.TXT /O:D $T DIR *.WKS /O:D
```

Let's assume you haven't turned off your computer, so the macro still is in memory. To save it as a batch file named DISPLAY.BAT, type:

```
DOSKEY /H > DISPLAY.BAT
```

Then edit the resulting DISPLAY.BAT file (use DOS 5.0's editor or your favorite word processing program) to get rid of everything except the following:

```
DOSKEY DOIT=DIR *.TXT /O:D $T DIR *.WKS /O:D
```

Save it as DISPLAY.BAT (remember to use text-only format). Now it will load the macro for you to use whenever you like!

TRAP KEEPING A COLLECTION OF YOUR FAVORITE MACROS.

Instead of sending your macros to a text file, extracting the lines you want, and saving them as separate batch files, you can direct all the macros you've created to a batch file. Here, the batch file is called MACROS.BAT (you can assign it any filename you like):

```
DOSKEY /H > MACROS.BAT
```

Each time you create a few more macros you like, direct them to your MACROS.BAT file using the /H switch, as in this command:

```
DOSKEY /H >> MACROS.BAT
```

Notice that you use >> rather than > to append them to the MACROS.BAT file, which already contains macros.

Next, edit MACROS.BAT to delete the lines you don't want and keep the lines you do want. All lines should begin with DOSKEY (so they'll all be macros). Then—this is the voodoo—call MACROS.BAT from your AUTOEXEC.BAT file so your favorite DOSKEY macros are in memory each time you start your computer. Assuming you've stored MACROS.BAT in your batch file directory, where you store all your other batch files (and put that in your path, of course), use these lines in your AUTOEXEC.BAT file:

```
DOSKEY
CALL MACROS
```

So, whenever you start up, your macros are automatically loaded in memory, along with DOSKEY.

If you don't want to call MACROS.BAT from your AUTOEXEC.BAT file but still want to have your favorite macros available at any time, just store MACROS.BAT in your \MACROS directory (or wherever you keep macros in your path) and type MACROS at any time to get them all in memory.

TIP ANOTHER WAY OF TURNING MACROS INTO BATCH FILES.

You also can use DOSKEY's /M (/MACROS) switch to save the macros themselves, but each line won't begin with DOSKEY because it's the command you create the macro with; it's not part of the macro itself. Enter the command this way:

```
DOSKEY /M > MACROS.DOC
```

Of course, specify whatever filename you want to use in place of MACROS.DOC.

TRAP WITH THE /M SWITCH, YOU MUST DO EDITING TO CREATE BATCH FILES.

To use macros that have been made into batch files with the /M switch, you have to edit the file and add DOSKEY (followed by a space) to the beginning of each line you want to be a command. And, of course, you have to save the batch file with a .BAT extension.

Here's an example of how a macro you saw earlier would appear in MACROS.DOC if you used the /M switch:

```
DOIT=DIR *.TXT /O:D $T DIR *.WKS /O:D
```

(Of course, if you had defined any other macros, they would be there, too.) By itself, DOIT makes no sense to DOS, so you need to add DOSKEY (again, followed by a space) to the beginning of the line if you want to use it as a line in a batch file.

TRAP YOU ALSO MAY NEED TO EDIT SPECIAL SYMBOLS.

You may need to do even more editing if you're converting macros to batch file commands with the /M switch. For example, $G in DOSKEY is equivalent to the DOS redirection symbol, >, and DOSKEY's $L is equivalent to DOS's <.

TIP USING DOSKEY TO DEVELOP YOUR BATCH FILES.

Use DOSKEY as you're developing batch files to test how they'll run. Test each command with DOSKEY on; then direct the commands to a file and turn it into a batch file. This method lets you see which line a problem is on as soon as it becomes a problem—then you don't have to guess how to debug a batch file that's several lines long.

TIP DELETING MACROS.

To delete a macro, type the following:

```
DOSKEY [macroname] =
```

No, there's nothing omitted from the end of the command. Just enter DOSKEY, the macro's name, an equal sign (=), and press Enter. To delete all your macros, simply press Alt-F10. ■

Kay Yarborough Nelson is the author of over a dozen computer books, including WordPerfect 5.1 Macro Handbook *(Sybex), and* The Little DOS 5 Book *and* The Little Windows Book *(Peachpit Press).* Voodoo DOS: Tips & Tricks with an Attitude *was released in March.*

DOS PART 1: PROGRAM IN DEBUG

Introducing Debug

Don't believe everything you hear. Debug is cryptic, and—used indiscriminately—it can cause problems. But it deserves a place in most every user's DOS directory.

by Dan Gookin

You've undoubtedly heard cautionary tales about DEBUG.COM: a novice who unwittingly destroys the boot sector on a floppy disk or experienced users, who should know better, accidentally reformatting their hard disks. The moral of such stories is clear: The average user should not meddle with Debug. But then you see it used in a magazine to create a handy-looking utility via something called a script file. So which is it, beauty or beast?

Both, really. Like most DOS commands, Debug can prove disastrous when used carelessly. And in a sense everything you've read is true. Debug *is* cryptic and powerful, and it *was* written with serious programmers in mind. But, the fact is, you don't have to be a pro to use Debug. Treated with proper respect, it is a useful tool with which you can create any number of quick-and-dirty little utilities—programs that let you reboot your machine without touching the reset button or change the status of your keyboard's Num Lock key at start-up. You even can use it to idly—and safely—cruise through memory or examine your disks up close.

DEBUG.COM: A set of DOS commands originally designed to assist programmers in finding and patching programs.

PROGRAM TYPE: External.

AVAILABLE: MS-DOS 1.25 and later. Updated in DOS 2.0 and again in DOS 4.0.

Over the next few issues of *DOS Resource Guide*, I'll present a series of articles aimed at demystifying Debug. You'll discover what a handy and curious program it is, learn a few interesting and relatively safe things you can do with it, and find out how to create Debug scripts and turn them into programs you can run at the DOS prompt. But before I introduce this month's elementary Debug experiments, you need a little background information.

Origin of the Species

Why is there Debug? Tradition, mostly. Microsoft and IBM wanted DOS to look a lot like the old, yet popular, CP/M operating system. CP/M had a debugging program called DDT—after the bug spray of the same name. To offer similar capabilities with DOS (and perhaps because IBM couldn't figure out if DDT had anything to do with CP/M's success), the companies decided to include a debugger in the first version of DOS. Debug was improved in Version 2.0 and overhauled again in Version 4.0.

Debug actually is a collection of powerful commands that have a direct pipeline to your computer's innards, which makes them potentially dangerous *to your* computer's health. The commands themselves are hardly mnemonic, consisting of one or two letters, but fortunately the list is short (see the table "Debug

Commands on page 38").

The commands have four primary duties:

- looking into memory addresses (called peeking) and changing information stored there (called poking)
- altering any program or data files stored on a hard or floppy disk, which makes it possible to fool with commercial software
- assembling and debugging programs
- changing the values in the registers of your central processor, or CPU chip

Registers are storage areas located within the CPU. They may contain a value that the CPU will need in the future—perhaps to compare to, add to, or subtract from values in other registers—or may hold an address that identifies where a particular bit of information is stored. Unfortunately, a register's name offers little clue to its purpose. Registers for Intel chips, for instance, come with such uninspired names as AX and SP.

Memory addresses may hold data, an application, or DOS. Each address may contain 1 byte (8 bits) of data, and each address location is represented by a four-digit number. This number is in hexadecimal notation, a counting system based on 16 rather than on 10, as our decimal numbering system is. I could go into a long-winded explanation of why this system works, but for now just nod your head and continue on (or refer to the sidebar "Taking the Hex Out of Hexadecimal" on page 36").

Peeking and Poking Through Memory

With that information under your belt, you're ready to get some firsthand experience. First, *invoke* Debug by typing the following command at the DOS prompt:

```
C:\DOS\DEBUG
```

This command assumes that Debug resides in your DOS subdirectory; change the pathname if you store Debug in another location.

Debug will greet you with its friendly hyphen prompt. (This is about as enlightened as Debug gets.) At the hyphen, you enter commands to tell it what to do. To see a list of commands on screen, press the question mark key and then Enter.

Debug can examine or change memory anywhere in your computer. But poking information into a memory location can be dangerous; you don't want to stick weird values into unfamiliar memory locations. That's referred to as poking around, and it may cause your computer to lock up, forcing you to reset it. (I'll tell you where to poke safely in a moment.)

To put Debug to work, try typing the following command at the hyphen prompt:

```
D
```

and press Enter. Debug then displays eight lines of text. The *D* stands for *dump*, an ungracious computer term that involves spilling out the contents of something. In this case, you're dumping 128 bytes of memory (see the figure "Memory Dump" on page 38).

Note that the display is divided into three sections. The first contains memory addresses; the second shows the 16 bytes of information stored at those addresses (all listed in hexadecimal, of course); the third shows the same 16 bytes, but this time in ASCII (text) format. When Debug encounters a character it can't display, it substitutes a period.

A computer with 1MB of memory has over a million memory addresses. Each of those addresses stores one byte of information, which Debug displays as two hexadecimal values separated by a colon. In the figure, for instance, the first address is B000:0000. The first hex

TIP A DIRECTORY BY ANY OTHER NAME

Here's an easy way to include several long subdirectory listings in a PATH statement, without exceeding the 127-character limit DOS places on an AUTOEXEC.BAT line. Before entering your PATH statement, use the SUBST command to shorten long directory names to a single drive letter. The following commands, for example, assign two complex pathnames to virtual drives E and F:

```
SUBST E: C:\LOTUS\BUDGETS\1991
SUBST F: D:\TAXES\TTAX\1991
```

Including E:\ and F:\ in your PATH statement after using these commands will point DOS toward the appropriate subdirectories. Doing so uses only six characters, however, rather than the almost 40 characters required by the actual directory names.

By default, DOS 3.0 and later recognize five drives at bootup. If you need to increase that number to accommodate additional SUBST commands, you can include a LASTDRIVE command line in your CONFIG.SYS file.

—J.N.

TAKING THE HEX OUT OF HEXADECIMAL

When you work in Debug, you need to work in hexadecimal, the base-16 counting system preferred by programmers as shorthand for the base-2 binary counting system that represents the on/off states of myriad switches inside a computer. Binary numbers, strings of ones and zeros, quickly become cumbersome to work with and awkward to display.

Consider, for example, the decimal number 100. In binary notation, it becomes 1100100. That's palatable, but the number 1999 decimal is 11111001111 in binary. It's getting uglier! How about the number 10,384: 10100010010000. Egads!

Hexadecimal notation makes things easier for programmers. Hex uses 16 digits: zero through 9 and then the letters A through F for values 10 through 15. The table of numerical equivalents compares decimal, binary, and hexadecimal numbers. Note that any four-digit chunk of a binary number can also be represented by a single hex digit. That's the logic behind using hexadecimal. So the number 100 is 1100100 binary but 64 hex, 1999 is 7CF hex, and 10,384 is more than manageable as 2890 hex.

—D.G.

NUMERICAL EQUIVALENTS

Decimal	Hex	Binary
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

value, B000, is known as the segment address and will differ from PC to PC. The second value, 0000, is the offset address.

You don't need to know the mechanics behind the segment/offset format for memory addresses; it's what the microprocessor uses, so we're all stuck with it. However, you should be aware that some segment and offset values are more interesting than others.

Peeking Around

To visit any memory location inside your PC, you simply type Debug's D command and specify the proper address. One of the more interesting addresses is at segment 40 hex (often written as 40h), offset 0. To see what's stored there, use the following command:

```
D40:0
```

This tells Debug to dump 128 bytes of memory starting at segment 40, offset 0 (memory address 0040:0000). To the untrained eye, the block of memory displayed when this command is invoked looks largely the same as the first block: boring. But memory address 0040:0000 holds many secrets.

For one, look at the ASCII section of your screen display and try to find the text that reads "D 4.0.:'0" or something similar. The *D* may be in upper- or lowercase, and the sequence may be split between two lines. Either way, what you're looking at is DOS's keyboard buffer, the place where the computer stores keypresses as you type them. The sequence mentioned above is the form in which the Debug command `D40:0` is stored in memory. Every other character in that area represents a keyboard scan code, which the PC uses to identify each keypress.

Other interesting tidbits of information are concealed at address 40:0, though they're not as obvious as the keyboard buffer. Most of the information stored is technical, and because it's all in hexadecimal notation, you can't easily make heads or tails of it.

Another interesting location, 40:4A, tells you how many columns are visible on the screen. To find it, perform a memory dump starting at 0040:0040 by typing:

```
D40:40
```

Then, count over 10 columns (A in hexadecimal notation), using zero as the starting point. The value stored at that location should be 50 hex. That same position in the ASCII section of the dump should be occupied by a *P*, because 50 in hexadecimal notation is 80 in decimal (16 x 5= 80), and, in ASCII, 80 is the value for *P*. This information tells you that your screen has 80 columns.

To look at a specific memory location, you can use variations on the D command. For example, typing in the following:

```
D40:4A L1
```

tells Debug "Dump memory at address 40:4A; list only *one byte.*" After pressing Enter, you'll see the single value 50 hex and the *P* in the ASCII column.

To prove that you have found the proper byte, try the following test, which works only on color displays. (If you're using a monochrome or Hercules display, skip down to the heading "Poking Around Video Memory.")

To quit Debug, type *Q* and press Enter. At the DOS prompt, change your screen width to 40 columns by typing the following:

```
C:\DOS\MODE 40
```

(assuming MODE.COM resides in a

subdirectory called \DOS). With the screen at 40 columns, return to Debug and type in the D40:4A L1 command again:

```
C:\DOS\DEBUG
D40:4A L1
```

The output from this dump command will "wrap" around to a second line, because Debug isn't formatted for a 40-column display. The value 28 hex (40 decimal) will be displayed instead of 50 hex. That's proof enough that memory address 40:4A holds the current screen width.

Next, quit Debug by typing *Q* and pressing Enter. At the DOS prompt, type this command:

```
MODE 80
```

to restore the screen to 80 columns. Then, start Debug again.

INSIDE MEMORY

Not all memory addresses are equally interesting to look at, and you can't always alter the information stored at a particular location safely. I've singled out a few places worth looking into using Debug's peeking capabilities and have noted whether you may poke in new memory values without fear of wreaking havoc.

Be forewarned, however, that for the most part "seeing" things in memory is possible only if you understand hexadecimal notation and can interpret the meaning of certain cryptic bytes. Exceptions are the final three areas in the table below. They contain text messages that you can read with ease.

—D.G.

Memory Locations	Contents	Safe to Poke?
0000:0000 to 003F:FFFF	Low-level data for DOS and the microprocessor	No
0040:0000	BIOS information	No
xxxx:0100*	Debug's memory "scratch pad"	Yes
B000:0000 to B000:0F9F	Monochrome video memory (text screen)	Yes
B800:0000 to B800:0F9F	Color video memory (text screen)	Yes
C000:0000	VGA BIOS copyright notice	No
C800:0000	Hard disk controller BIOS	No
F000:0000 to F000:FFF0	Text messages about the BIOS	No
F200:0000	BIOS copyright message (on some systems)	No
FFFF:0000	BIOS copyright date	No

* Address varies according to what is loaded in memory.

Poking Around Video Memory

One of the safest places to poke around is in video memory. It's unlikely that you'll cause any damage, and changing the bytes stored there can create interesting visual effects. Of course, before you can change anything, you need to know the screen's address in memory.

If you have a monochrome display, the screen starts at memory segment B000 hex. If you have a color display (CGA, EGA, VGA, and so on), the screen starts at segment B800 hex. Because most people have color monitors, the following examples use B800 hex; if you have a monochrome display, remember to substitute B000 hex instead.

You use Debug's E command to poke bytes or text into memory. As with the D command, you type *E* and then a memory address, following up with the new value you want to poke there. If, for example, you type this command:

```
EB800:780 1
```

and press Enter, you'll see a little happy face in the left-most column, near the middle of the screen. This indicates that the 1 character, the happy face, was poked into video memory by the E command. Now try the following:

```
EB800:780 "HELLO THERE!"
```

Oops! The message "HLOTEE" is colorful, but it's probably not what you were expecting. It turns out that video memory contains two bytes for each character displayed: The first byte is the character value—its numerical code—and the second is the character's color code, or attribute. To prove this, try typing the following sequence instead, being sure to use a lowercase *p*:

```
EB800:780 "HpEpLpLpOp pTpHpEpRpEp!p"
```

Now, you'll see the message "HELLO THERE" displayed—in inverse type. This happens because the code for *p* is 70 hex, which tells the video hardware to display the characters as black text on a white background. The above line also shows that instead of puzzling over hex bytes, you can simply enclose a text sequence in double quotes, as long as you insert an attribute code after every character. (If you want to know

DEBUG COMMANDS

Command	Description
A	Assembles 8086/8087/8088 mnemonics.
C	Compares two portions of memory.
D	Displays the contents of a portion of memory.
E	Enters data into memory, starting at a specified address.
F	Fills a range of memory with specified values.
G	Runs the executable file that is in memory.
H	Performs hexadecimal arithmetic.
I	Displays a 1-byte value from a specified port.
L	Loads the contents of a file or disk sectors into memory.
M	Copies the contents of a block of memory.
N	Specifies a file for an l or w command, or specifies the parameters for the file you are testing.
O	Sends a 1-byte value to an output port.
P	Executes a loop, a repeated string instruction, a software interrupt, or a subroutine.
Q	Ends a Debug session.
R	Displays or alters the contents of one or more registers.
S	Searches a portion of memory for a specified pattern of one or more byte values.
T	Executes one instruction and then displays the contents of all registers, the status of all flags, and the decoded form of the instruction that Debug will execute next.
U	Disassembles bytes and displays the corresponding source statements.
W	Writes the file being tested to a disk.
XA	Allocates expanded memory.
XD	Deallocates expanded memory.
XM	Maps expanded memory pages.
XS	Displays the status of expanded memory.

more about attribute codes, consult *The New Norton Programmer's Guide to the IBM PC and PS/2* from Microsoft Press.)

Using the Fill Command

Debug has several commands that examine the contents of memory. So far, I've touched on two of them: D, which dumps the contents of memory; and E, which pokes new bytes into memory. You can use another memory command, F, to create interesting screen effects by filling a block of memory with a particular value and attribute.

To get a sense of how this works, type this command:

```
FB800:0 LFA0 "A"
```

Kinda psychedelic, huh? This command tells Debug to fill in the bytes starting at address B800:0. The number of bytes to fill—the length—is FA0 hex, or 4000 decimal. The value provided within quotes does double duty: It tells Debug what character code (A) to place in the blocks specified and what attribute code to use (41 hex). The value 41 hex corresponds to blue text on a red background. A lowercase *p* makes things more legible. To prove it, try:

```
FB800:0 LFA0 "Ap"
```

This time *A* functions as just the character code; *p* is the color code.

Now, try one more variation:

```
FB800:0 LFA0 "!" CE
```

This time, an exclamation mark is the character code to be inserted and the attribute value appears outside the quotes. There is no corresponding attribute character for this particular code, so you must use a hexadecimal value. You end up with a screenful of bright-yellow flashing exclamation points on a red background. You can experiment further with this command using values C1 through CD and CF in place of CE. Now who says Debug is boring?

Second Time Around

In the next issue, I'll tell you about Debug's disk access, how it can create, save, and load files, as well as edit "raw" information on disk. For now, type *Q* and Enter to exit Debug, then type CLS before you have a brain seizure. ■

Dan Gookin is a free-lance author living in Issaquah, Wash. His upcoming book, Dan Gookin's Book of Underground DOS *(Bantam) explains Debug and other cool stuff at great length.*

MEMORY DUMP

Memory Addresses · **Bytes (in hexadecimal)** · **Bytes (in ASCII)**

```
B000:0000  4D 01 B0 06 01 00 78 00-41 4E 53 49 00 00 78 00   M.....x.ANSI..x.
B000:0010  00 00 39 03 53 C0 C1 00-CC 00 43 4F 4E 20 20 20   ..9.S.....CON
B000:0020  20 20 51 10 70 01 70 01-70 01 76 03 36 04 70 01     Q.p.p.p.v.6.p.
B000:0030  CA 04 F6 04 F6 04 70 01-70 01 70 01 70 01 70 01   ......p.p.p.p.p.
B000:0040  70 01 70 01 70 01 70 01-1A 0A 41 5E 06 42 66 06   p.p.p.p...A^.Bf.
B000:0050  43 4E 06 44 59 06 48 20-06 4A DA 06 4B 17 07 52   CN.DY.H .J..K..R
B000:0060  50 07 66 20 06 68 57 07-6C 53 07 6D AA 06 6E 2B   P.f .hW.lS.m..n+
B000:0070  07 70 94 07 71 7B 06 73-98 06 75 A1 06 00 00 00   .p..q{.s..u.....
```

Segment Address **Offset Address**

TAKE THE DREAD OUT OF DOS!

If you're trying to succeed with DOS, but feel like a General Failure...

DOS RESOURCE GUIDE IS WRITTEN FOR YOU

DOS RESOURCE GUIDE, the quarterly periodical packed with tips, tutorials, and information on all aspects of PC computing, is your antidote for computing confusion.

DOS RESOURCE GUIDE #1

Make DOS Work for You

Covers six areas of computing that are vital to DOS users: start-up files (AUTOEXEC.BAT and CONFIG.SYS), batch file creation, hard drives, safe backups, organizing file management, and hardware maintenance. Includes: List'em, a batch file that helps you keep appointments; 10 tips to keep your hard drive healthy; speed-up tips for a logy hard drive; a batch file that automates backups; DOS shortcuts, hardware dos and don'ts. Plus 42 provocative pointers on DOS.

DOS RESOURCE GUIDE #2

Guide to Windows Software

SOLD OUT

A compendium of software briefs that tells you what's up in the world of Windows. (Even a DOS [illegible] keep up with the competition.) Guide #2 briefly reviews over 140 programs for Microsoft Windows, including word processors, communications programs, financial and mathematical programs, management programs, developer's tools, and graphics software. Complete list of manufacturers.

DOS RESOURCE GUIDE #3

The Essential Guide to DOS

Who are we to argue with success! Our second edition of this classic is updated to include information on DOS 5. Guide #3 includes an annotated list of DOS files and when to use them; articles that will change your view of CHKDSK, COPY, and XCOPY; a cluster of batch file assists; and some clear-headed advice on organizing files on your hard drive. Plus 14 titillating tips on DOS.

DOS RESOURCE GUIDE #4

The Complete DOS Q&A

Breaks DOS into functional sections: Fundamentals, Set-up, Improving Productivity, Memory Management, Hardware Upgrades, Batch Files, and Using Windows. Each section is followed by a question and answer session on the topic. Plus, seven short programs to protect your data; do-it-yourself memory, advice on CONFIG.SYS and AUTOEXEC.BAT, and beginner's Basic. Not to mention nine tips for the DOS timid.

So, If DOS keeps asking you those hair-raising questions, like "Are you sure?"... **THEN** take a look at a few back issues of DOS RESOURCE GUIDE, and take the dread out of DOS!

Use our 800 number and have your credit card ready!

1-800-238-6580

Back issues of DOS RESOURCE GUIDE are $5.50 post-paid!

Or send your check or money order to:

DOS Resource Guide
P.O. Box 3162
Salisbury, MD 21802

(Please indicate what issue(s) you are purchasing.)

DON'T MISS AN ISSUE!

Upgrade Now!

Microsoft's latest DOS version is packed with appealing features. Here are a half-dozen reasons why you should switch.

by Jack Nimersheim

Let's face it, we all share a love/hate relationship with MS-DOS. It allows us to accomplish a lot with our personal computers, but the effort required to realize these accomplishments can, at times, seem Herculean. And then, just about the time we feel we've mastered DOS, Microsoft goes and changes it.

I confess, I never upgraded to DOS 4.x. In my mind, the end simply didn't justify the means. I could live without its support for larger hard disks; to be honest, I believe organizing files using smaller storage units is generally more efficient. Furthermore, DOS 4.x's shell was an even less impressive attempt at providing a graphics-based DOS interface than were early versions of Windows. Finally, although DOS 4.x did provide support on certain systems for the expanded memory specification (EMS) 4.0—a feature that interested some people—Quarterdeck Office Systems' QEMM memory manager, which I had been using for years, did so more reliably and with greater ease.

After analyzing the strengths and weaknesses of the "new and improved" DOS 4.x, I saw no overriding reason to abandon good old reliable DOS 3.3. And I've never regretted that decision.

Then, slightly more than a year ago, Microsoft released MS-DOS 5.0. There I was, facing the same question all over again: to upgrade or not to upgrade? I said yes, and have no regrets this time around, either.

PRODUCT INFORMATION

MS-DOS 5.0
Price: $99.95
Microsoft Corp.
1 Microsoft Way
Redmond, WA 98052
206-882-8080

DOS 5.0 offers something for everyone. If you use a single-floppy, 8088-based PC, you'll appreciate its redesigned DOS shell. Those with more powerful PCs will revel in the extra memory this new DOS version makes available to application programs—often 50K or more of conventional RAM on 286 and higher systems. In addition, DOS 5.0 provides a safety net that protects against any compatibility problems you might encounter with programs that ran perfectly well under earlier DOS versions. In short, this new DOS covers a lot of ground. I'll highlight just six of the many reasons why *you* should consider an upgrade.

1 Conviviality

I don't know about you, but I get nervous whenever I contemplate installing a new operating system on my PC. My biggest concern usually involves compatibility—or, more correctly, a lack thereof. Will the application programs on which I depend work reliably under the new DOS release? In the past, you had no guarantee that they would. MS-DOS 5.0's new SETVER command all but eliminates this potential problem.

The issue of incompatibility arises because some applications check to see which release of DOS you're using. Generally, this is done to make sure you don't try to run a program under an older

DOS version that doesn't support the specific functions the application requires to work properly. (This is why the system requirements listed for a program generally include a statement such as "For DOS 2.1 or higher.") Sometimes, however, this strategy can backfire, causing a particular program to refuse to load under a DOS version later than the highest one for which the program checks. Using the SETVER command, you can "fool" an older application into thinking it's running under an older DOS release than Version 5.0.

For example, I installed MS-DOS 5.0 on my system while I was writing a book about GeoWorks Ensemble. The first time I tried to load Ensemble under DOS 5.0, my system spun off to electronic limbo. It locked up so badly I had to perform a warm boot by pressing the reset button. Clearly, Ensemble had trouble running under Version 5.0.

To eliminate the problem, I turned to SETVER, a command made available by a `DEVICE=C: \DOS\SETVER` line in my CONFIG.SYS file (thanks to DOS 5.0's installation procedure, which added the line automatically).

When DOS 5.0 encounters the SETVER command during system setup, it creates a version table, which the program uses to cross-check valid release numbers for several popular applications. Then I typed the following:

```
SETVER GEOS.EXE 4.01
```

to add a new listing for GEOS.EXE (the program that loads Ensemble) to the original SETVER table. You can incorporate new applications into your SETVER table using the following command syntax:

```
SETVER [executable program name] [DOS version number]
```

One caveat: Because SETVER is loaded from your system's CONFIG.SYS initialization file, you'll need to restart your system before any modifications made to the version table will take effect. (For details on how to use SETVER, see "A Farewell to Incompatibility" on page 19.)

2 A Handy Escape Clause

It takes a big person to admit he or she is wrong, and it takes a big company to admit its product may not be right for everyone. I give Microsoft credit for having the chutzpah not only to admit that the IBM-compatible market is so splintered that its newest DOS simply may not work on some systems, but also for providing an easy way for users to dump DOS 5.0, if it doesn't run well on their PCs.

As part of its installation process, DOS 5.0 creates a special "uninstall" disk. It then transfers to this disk certain files from the DOS version it's replacing, along with critical information about your system's current setup: your CONFIG.SYS and AUTOEXEC.BAT files, the contents of your hard disk's root directory, a copy of your boot disk's file allocation table (FAT), and other things of this sort.

If, after installing MS-DOS 5.0, you decide you don't like it (or if you have trouble running the new operating system on your machine), you can simply boot your PC from your uninstall disk. You'll be asked if you want to return to your previous setup. If you answer yes, DOS 5.0 will remove itself from your hard disk, restore the earlier DOS version, and return all the information it copied to the uninstall disk during installation. How's that for a safety net? Of course, you're still out $100 or so.

If you do decide to stick with DOS 5.0, you'll be glad you did. It has a lot to like, including the ability to work with more memory than previous versions.

3 More Memory at Your Service

So much has been written about DOS 5.0's improved memory-management capabilities, you're probably sick of hearing about them. So, I'll keep this brief.

By far, the most convincing argument for upgrading to DOS 5.0 is its ability to load portions of DOS into the high memory area (HMA)—the 64K block of RAM just above the 1MB address at which extended memory begins. Storing DOS in the HMA can give you anywhere from 45K to more than 60K of additional conventional RAM—that is, memory below 640K—in which to run application programs. How much extra memory you'll have depends on the type of computer you own.

TIP CONSERVING MEMORY

One way to increase the memory available to your system is to remove the nine interrupt stacks that DOS automatically sets up on 80286-based and higher machines. Adding the following command:

```
STACKS=0,0
```

to your CONFIG.SYS file frees approximately 2K of conventional RAM for application programs.

A word of caution, however: Removing the interrupt stacks can cause trouble in some 80386 and 80486 systems that are running Windows 3.0 in Enhanced 386 mode. Eliminating the STACKS=0,0 statement from CONFIG.SYS should resolve the problem.

—J.N.

PC/XT owners—those whose computers contain a first-generation 8088 or 8086 CPU—are pretty much out of luck on this one. DOS 5.0's advanced memory-management features kick in on 80286-based systems, or so-called AT machines. They really shine, though, on a 386 system or higher.

Note: The following examples apply to a directory system that stores the various DOS memory-management utilities in the main, or root, directory of drive C. If these files reside in a different directory on your system, be sure to substitute a pathname containing the appropriate drive and directory designation so that DOS can find them.

To access the HMA on an 80286-based PC, the first thing you should do after installing DOS 5.0 is add the following two lines to the beginning of your CONFIG.SYS file:

```
DEVICE=C:\HIMEM.SYS
DOS=HIGH
```

The first line in this sequence loads HIMEM.SYS, which prepares the HMA to accept portions of the DOS code. The second line tells DOS to load itself into the HMA. These two lines alone increase by approximately 45K the amount of conventional RAM available to your applications.

Anyone who has a 80386 or 80486 system can reclaim even more conventional RAM from DOS. The CONFIG.SYS file on my own 486-based PC, for example, begins with the following three lines:

```
DEVICE=C:\HIMEM.SYS
DEVICE=C:\EMM386.EXE RAM
DOS=HIGH,UMB
```

The second line in this sequence loads the DOS 5.0 expanded-memory manager (EMM386.EXE), which converts the 4MB of extended memory installed on my system into expanded memory (EMS 4.0). Including the RAM parameter in this line and adding the UMB switch to the `DOS=HIGH` command lets me use DOS 5.0's new DEVICEHIGH and LOADHIGH statements to load my mouse driver, along with several memory-resident programs, into the otherwise wasted RAM located between the 640K and 1MB memory addresses.

By way of comparison, after starting my system with DOS 3.3, I used to have only a little more than 570K of conventional RAM left. Using all the memory-management tricks available in DOS 5.0, I've increased that number to 621K. The extra 50K comes in mighty handy for running today's RAM-hungry applications.

4 Blunder-Proofing

MDOS 5.0 not only is generous, it's also much more forgiving than its electronic ancestors. Let me give you a somewhat embarrassing example of what I mean.

Just last week, I accidentally erased an entire directory of files from my hard disk. Not even 10-year DOS veterans, it appears, are above committing an occasional major blunder. With previous DOS releases, I would have had to rely on a utility program, such as Central Point Software's PC Tools or Symantec's The Norton Utilities, to salvage my work. Not with MS-DOS 5.0. Its UNDELETE command—actually, a licensed version of PC Tools' file-recovery program—came to my rescue.

A new UNFORMAT command is equally ready to jump into the fray, should you commit an even more destructive faux pas—such as accidentally reformatting a hard disk. (And, yes, I've been known to do that a time or two also.)

5 A Fresh New Face

Honesty forces me to confess that I don't perceive graphical user interfaces, or GUIs, as being the Holy Grail many computer journalists and industry analysts proclaim them to be. I still spend most of my time working in standard DOS programs, although I favor a couple of capable Windows applications for working on specific projects. Still, I must admit that the DOS 5.0 shell, DOSSHELL.EXE, far surpasses the anemic efforts Microsoft invested in designing the graphical shell included with DOS 4.x.

For one thing, this new DOS shell is a genuine GUI, with the emphasis on the *G* in that acronym. Basic activities, such as changing disk drives, formatting disks, copying files, and the like, are transformed into true point-and-click procedures, although you don't absolutely need a mouse to use Version 5.0's DOSSHELL. And unlike DOS 4.x's shell, which relied heavily on text symbols to simulate a graphics-based screen (remember the DOS 4.x directory tree?), this DOS shell uses bit-mapped

TIP ANALYZING YOUR ARCHIVES

If you need to see a list of the files on a particular disk that were archived using the DOS BACKUP command, simply invoke RESTORE and add a /D parameter. DOS displays the appropriate filenames without initiating restoration procedures, and you can use wildcards to specify that only certain filenames be displayed. The following command, for instance, displays a list of all files with a .TXT extension that reside on drive A, without actually restoring the files on drive C:

```
RESTORE A:*.TXT C: /S /D
```

Note that adding the /S parameter instructs RESTORE to search all subdirectories on the backup disk in drive A and to include all files it finds that match the specified search criteria in the resulting file list.

—J.N.

graphics that take full advantage of your monitor, even if it's a high-resolution VGA display.

The most impressive aspect of the revamped DOS shell, however, is its newfound ability to work with several applications concurrently, using its so-called Task Swapper. As its name implies, this features means you no longer have to quit an application, open another one, then shut that one down to return to the first one. Although I still use Quarterdeck's DESQview for complicated projects that require true multitasking, DOS 5.0's Task Swapper comes in handy when I need to juggle a couple of applications to get some quick-and-dirty work done.

6 Better Basic

For those with a daring spirit, DOS 5.0 provides the perfect introduction to programming. Admittedly, not everyone is interested in designing applications. If you are, however, you'll find that QBasic, the new Basic language included with DOS 5.0, is a heck of a lot friendlier than GW-Basic, the programming language Microsoft bundled with earlier DOS versions.

On-line help messages and easy-to-use pull-down menus make QBasic an ideal starting point for someone just beginning to explore the wondrous but mysterious world of PC programming. Its "intelligent" editor detects syntax errors as they appear in each line of code and even provides suggestions on how to correct them.

Although QBasic doesn't require that you organize your programs with line numbers, as did GW-Basic, it does recognize numbered code. That's good; I don't have to pitch all those personal utilities I cobbled together back when the Basic bug first bit me. Heck, QBasic's so easy to use, I might just hack together a few more, as the mood hits me.

If you decide you're not the programming type, don't be quick to remove QBASIC.EXE from your DOS subdirectory. QBASIC.EXE works hand-in-hand with EDIT.COM, the file containing DOS 5.0's fine replacement for the Edlin line editor. At last, Microsoft has devised a straightforward command line editor that lets you catch simple misspellings or dash off a quick note. Although the DOS 5.0 editor won't replace your word processor, its pull-down menus, cut-and-paste and search-and-replace capabilities, and extensive on-line help alone are worth the price of this upgrade.

The Final Verdict

It's never easy to decide whether to buy the latest release of a previously reliable operating system. I know a lot of people who were taken in by the initial hype surrounding DOS 4.x—a product that, as it turned out, offered more promise than power and, initially at least, contained more bugs than an entymologist's laboratory. Many of these same folks are understandably suspicious of DOS 5.0. If anything, this new version is touted even more loudly than DOS 4.0. In DOS 5.0's case, however, the enthusiasm is justified. I've been using this new DOS for more than a year, and I've never once regretted my decision to replace DOS 3.3 with this latest in a long line of DOS upgrades.

Is MS-DOS 5.0 perfect? Of course not. It would be nice, for example, if I could swap tasks directly from the command line, instead of first having to open the DOS shell. But I started out asking, "Do I upgrade, or don't I?" My answer remains an enthusiastic "Yes!" ■

TIP MORE IS LESS

If you frequently use the DOS TYPE command to view text files stored in standard ASCII format, the following batch file, which you can name VIEW.BAT, will simplify your life:

```
CLS
TYPE %1 | MORE
CLS
```

The CLS command in the first line simply clears the screen before the TYPE command begins running. (The final CLS command accomplishes the same thing at the end of the file, before returning you to the DOS system prompt.) The MORE parameter in the batch file's second line causes DOS to pause at the end of each full screen of text, giving you the opportunity to read what is on the screen. Pressing any key tells DOS to display the next section of your file.

After creating and saving VIEW.BAT as you would any batch file, you put it to work by invoking the batch file and specifying the name of the file you want to display (%1). For example, to view the contents of a file called MYTEXT.ASC—one screen at a time—you would type:

```
VIEW MYTEXT.ASC
```

—J.N.

Computer journalist Jack Nimersheim has been using and writing about personal computers for more than 10 years. His First Book of MS-DOS *(Prentice Hall Computer Publishing) spent 12 months on Waldenbooks' best-seller list.*

Understanding Serial Ports

Serial transmission works flawlessly—most of the time. But it pays to know about potential problems and the appropriate solutions before conflicts arise.

by Brett Glass

A serial port is one of the most useful—and most misunderstood—interfaces you can attach to an IBM-compatible computer. When it works (and under most circumstances it does), a serial port provides an easy and relatively foolproof way to connect a modem, a mouse, a laptop or palmtop computer, or a serial printer. But many users run into unexpected difficulties and conflicts when they attempt to use more than two serial ports on a PC. And when a serial port is configured incorrectly (some PC compatibles, especially no-name clones, even come from the store this way), crashes, lockups, or even physical damage to the machine can result.

How can you avoid these problems? How does a serial port work? How can you make the most of the serial ports on a machine running under DOS? Under Windows? What do you do when you run out of serial ports? The answers to these questions are essential knowledge for any PC user.

What Is a Serial Port?

A serial port gets its name from the way it transmits and receives data: serially, or one bit at a time. A parallel port, by contrast, sends 8 bits at the same time. A serial port sends information more slowly than a parallel port, but data transmission is simpler. While a parallel port requires eight wires for data (plus more for control signals), the most basic kind of serial port needs only three wires: one to send data, one to receive data, and one—called a common—to act as a ground. A fancier serial port of the type you're likely to find on a PC compatible has nine wires—the three just mentioned plus six more that exchange control signals with a modem. When you see a serial port on the back of a PC or a modem, it usually has a connector like one of those shown in the figures accompanying the sidebar, "Serial Port Connectors," on page 48.

An IBM compatible's serial ports are called asynchronous, because they don't transmit data continuously to the beat of a clock signal, as synchronous serial ports do. Instead, its ports start to send information only when they have something to transmit; they stop as soon as they complete the transmission. Each character (which can be anywhere from 5 to 8 bits long but usually comprises 7 or 8 bits) is preceded by a *start bit*, a warning that a character is coming, and followed by a *stop bit*, which signals the end of the character.

Typically, a serial port on an IBM compatible is controlled by a chip called a UART, or universal asynchronous receiver and transmitter, which conveys data from the computer's CPU to the serial port and vice versa. The first IBM PC used a UART called the 8250B, which was made by National Semiconductor Corp. Later models, such as the 80286-based AT, use a more advanced chip, the 16450A, and the PS/2 and the most powerful clones use an even more powerful chip called the 16550A. The latter has some special advantages I'll cover later.

A Confused Design

Managing serial ports on PC compatibles would be simple had the original designers done a better job of organizing their work. Unfortunately, the serial communications facilities of the original PC apparently were the work of several designers, who did not even agree on such basic issues as the maximum number of ports the machine could have. One portion of the IBM PC's BIOS (basic input/output system, software instructions stored in the PC's hardware), for example, contained a routine that a program could call to get a brief list of the peripherals installed in the machine. The routine could return a number from zero to 7, which implies a maximum of seven serial ports were possible. But another part of the program—which took inventory when the computer was started—checked for only two. And another section—a table of available serial ports maintained by the BIOS—had room for the addresses of four ports. Finally, IBM's hardware manuals specified addresses for only two serial ports: COM1 and COM2 (see the table "Serial Port Addresses and Interrupts"on page 46), and these were the only two ports early versions of MS-DOS supported.

As the PC became popular, programmers quickly discovered that the routines included in the BIOS and DOS were not fast enough or capable enough to handle many kinds of communications. Both used *polled* input and output (which requires a program to check the serial port constantly to avoid missing incoming characters) rather than *interrupt-driven* input and output (which lets the machine handle characters immediately). This meant that a program that used the BIOS missed characters if it did a disk access, screen update, or any other operation that prevented it from checking the port every 1 or 2 milliseconds. As a result, programmers who created communications programs often had to start from scratch, writing their own code to manipulate the hardware directly.

Recognizing that polled input and output was a poor solution for serial communications, IBM and other vendors soon revised the communications portion of the BIOS to use interrupt-driven input and output. Only very early IBM PCs and compatibles used polled I/O. The switch to interrupt-driven I/O was a much needed improvement, but the other limitations remained.

Although few users of the IBM PC and compatibles needed seven serial ports, many needed more than the two that IBM's hardware specifications allowed. Responding to this need, Quadram and AST—the two most popular board vendors in the early days of the PC—created de facto standards for two more serial ports: COM3 and COM4. DOS was even updated to accommodate four serial ports.

If you compare the the interrupt settings for COM3 and COM4 with those for COM1 and COM2, it doesn't take much scrutiny to see a big problem: They use the same interrupts as COM1 and COM2, but interrupts on the IBM PC bus were not meant to be shared. In fact, a quick look at the circuit diagram of any IBM compatible shows that if two expansion cards attempt to share an interrupt level, a conflict will arise that may result in interrupts going unrecognized by either card or that may even cause one card to send potentially damaging electrical signals to the other. The Extended Industry Standard Architecture (EISA) bus was designed to allow cards to share interrupts, but this feature works only if all the cards installed in a system are EISA cards; virtually all serial port cards are of the AT type, or ISA cards.) Another potential conflict also exists. Because IBM didn't reserve the I/O addresses for the de facto serial port standards, some peripheral boards may use them for other things. For instance, certain VGA boards with Windows accelerators—including the STB Wind/X board—may have I/O addresses that

TIP FOOLING FORMAT

Even with all its capabilities, DOS still can be quite dumb. For example, you usually need to enter a complicated set of command parameters to the FORMAT command each time you want to format a 720K 3.5-inch disk in a 1.44MB drive. A simpler alternative is to use a standard text editor and create the following batch file, which you can call BFORMAT.BAT:

```
CLS
FORMAT A: /N:9 /T:80
MENU
```

Each time this batch file runs, it automatically formats a 3.5-inch disk in drive A to 720K. Using the /N and /T parameters guarantees that this command will work properly, regardless of which version of DOS is installed on your PC.

—J.N.

conflict with the de facto COM4.

The existence of so many IBM-compatible computers with three or four serial ports might well prompt you to ask why such machines aren't forever blowing chips or crashing. The reason is simple: For an interrupt conflict to cause problems, two serial ports must be trying to use the same interrupt at the same time. For example, if COM1 is using interrupts (e.g., for a communications program) while COM3 isn't (if it is driving a serial printer via the MODE command, for instance), no conflict will arise. But if your mouse driver program, which controls how your mouse and your PC communicate with one another (and which one uses interrupts), is using COM3, your machine might hang when you move the mouse, since COM3 uses the same interrupt as COM1.

SERIAL PORT ADDRESSES AND INTERRUPTS

IBM Standard

Port	Interrupt	Hexadecimal I/O Addresses
COM1	IRQ 4*	3F8-3FF
COM2	IRQ 3	2F8-2FF

De Facto Standard

Port	Interrupt	Hexadecimal I/O Addresses
COM3	IRQ 4	3E8-3EF
COM4	IRQ 3	2E8-2EF

*IRQ stands for interrupt request.

Making Peace

If your machine currently has more than two serial ports, you can try several things to avoid conflicts. The first, and most risky, is to configure your software so that two serial ports never try to use the same interrupt at the same time. For instance, if you are using a mouse driver for a mouse connected to COM1, you should configure other programs that use interrupts, such as communications programs, to use either COM2 or COM4, avoiding COM1 and COM3, where a conflict could occur. This approach is dicey, however, because it isn't always easy to tell whether a program uses interrupts.

A better way to avoid conflicts is to deviate from the de facto standard a bit. If your serial card allows it, use a different, unused interrupt (for example, IRQ 5 or IRQ 2) for COM3 or COM4. Then, tell your modem program the interrupt level each port is using. (Not all modem programs allow this, but the best ones do.)

A third way is to buy a specially designed multiple-serial-port card, such as those made by Arnet Corp. (615-834-8000) and Digiboard (800-344-4273). These boards aren't compatible with all software but do work with many programs. The best solution of all, however, is connecting no more than two serial devices to your PC, using COM1 and COM2, and to find other ways to attach additional devices.

One way to free up serial ports on your system is to use parallel ports instead. If you have a serial printer, consider refitting it with a parallel interface or buying a parallel-to-serial converter—a device that converts the output from a parallel port to serial data. These converters, which cost as little as $35 to $40, often another advantage, as well: Many also contain a hardware print spooler, which, like a software print spooler, allows printing to go on while you continue to use your computer for other tasks. A spooler temporarily diverts data intended for your printer to memory (RAM) or disk, which can accept the data much faster than the printer, thus returning control of your computer to you very quickly after you start a print job. A hardware print spooler, sometimes called a *print buffer*, contains its own RAM and thus uses none of your computer's resources. A software print spooler sets aside a portion of your computer's own memory or hard disk to act as a temporary holding area for printer data. The software spooler then sends data to the printer during moments when the computer is otherwise idle, such as between keystrokes. (*Spooler* is an acronym derived from *simultaneous print operations on-line*.) A hardware print spooler has two key advantages over a software print spooler. First, it's guaranteed to work with every application program (some software print spoolers won't) and, second, it doesn't take up any RAM in your machine (the converter has its own RAM). If you're thinking of using a parallel-to-serial converter, however, be aware that such devices generally are designed to transfer data only one way: from the parallel port on your machine to a serial output device. It won't work with devices—such as modems—that require characters to flow both to and from your computer.

If you're currently using a serial port LAN (either because you're working on a laptop or because all of the expansion slots in your system are filled), you still can get LAN capabilities via a parallel port. Two companies—Xircom Inc. of Woodland Hills, Calif., and IQ Technologies of Bellevue, Wash.—make parallel-port LAN adapters for ethernet, Arcnet, or Token Ring setups. These products offer the advantages of a standard LAN protocol, as well as better performance and one more serial port.

If you use a laser printer through a serial port, you have a great deal to gain by switching to parallel operation. Typically, your rate of throughput will improve better than 20 times, which makes it worthwhile to switch even if you're not running out of serial ports. You'll experience similar speed improvements if you use a laptop file-transfer program—such as Traveling Software's LapLink—through a parallel port rather than a serial port.

Parallel ports have two limitations you should know about, however. First, IBM compatibles are designed to support at most three parallel ports; if you don't plan carefully, you may find yourself having to work around *this* maximum. Second, parallel ports allow a shorter maximum cable length than serial ports: The practical limit is 15 feet to 25 feet as opposed to 3,000 feet or more for serial lines. (Of course, if you use a parallel-to-serial converter, the range will be the same as for any other serial port.)

To make the most of the serial ports you have, you can use a switch box to connect several peripherals to a single serial port. This technique works well if you know that you won't be using certain devices simultaneously—a mouse and a laptop file-transfer cable, for instance. If you're especially lazy, you even can buy an "intelligent" switch, an electronic dispatcher that can connect the serial port to one of several devices on command. These switches, normally used with printers, often are remarkably flexible; some contain spoolers and facilities to convert between serial and parallel data.

Be careful, however. If you use a switch box to make a serial port available for multiple purposes, make sure that the programs that use the port—perhaps a laptop file-transfer program and a memory-resident mouse driver—don't fight with one another for control of the port. The ensuing struggle sometimes can lock up your machine. The best way to avoid such problems is to have only one of the programs loaded at a time.

Another way to remove a peripheral from a serial port is to determine if there is a dedicated interface card available for the device. If available, the card can be plugged into one of your computer's expansion slots and the device plugged into the card, freeing up a serial port. For instance, to free a serial port currently used by your mouse, you can get a *bus mouse* that comes with its own interface card. Bus mice cost about the same as serial mice, so if you have an empty slot in your computer, this is an easy way to save a serial port.

Using a LAN card that plugs directly into your PC's bus may dramatically improve network performance, as well as freeing up a port

TIP FASTER TASK SWITCHING

DOS 5.0's task switching capability is a handy feature. Unfortunately, however, going from one task to another takes a bit too much time. But if your computer is equipped with expanded or extended memory, you can use a RAM disk (also called a virtual disk) to speed things up.

Start by setting up a RAM disk by adding a line such as the following to your CONFIG.SYS file:

```
DEVICE=C:\DOS\RAMDRIVE.SYS 512 /E
```

Then add the following line to your AUTOEXEC.BAT file:

```
SET TEMP=D:\
```

(The above commands assume that RAMDRIVE.SYS is in the DOS directory on drive C and that DOS will assign drive D as the specification for your RAM drive.) When you reboot your computer, the new settings will take effect, and DOS will switch you from one application to another more quickly.

To understand why this works, it helps to know a little more about task switching. The DOS 5.0 task switcher normally uses your computer's hard disk as a repository for information about the tasks it "puts to sleep" when you switch from one application to another. To do this, it creates a temporary batch file for each task on your active task list. These batch files contain information the task swapper uses to locate the programs and data files named in your active task list. It also creates a larger temporary file called DOSSHELL.SWP that contains other information required to swap tasks.

When DOS needs to create a temporary file, it first looks in the DOS environment space to see if it contains a variable named Temp. If DOS finds a valid pathname assigned to Temp, it will write all temporary files to the drive specified by the pathname. By requesting that DOS write its temporary files to memory, which is faster than accessing a hard drive, you ensure speedier task switching.

Although 512K is a good starting size for your virtual disk, you may need to experiment to maximize performance gains and minimize memory usage.

—J.N.

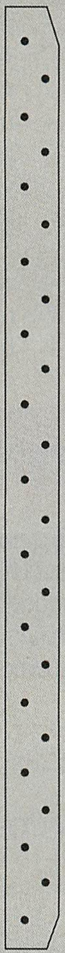
SERIAL PORT CONNECTOR

The industry-standard serial port connector follows a standard called RS-232C, published by the Electronic Industries Assoc. (EIA) in 1969. The standard defines 25 pins, most of which are rarely used.

If you'd like to see one of these connectors, the best place to look is the back of a modem. Chances are you'll find a female 25-pin socket that's intended to connect to the serial port on a computer or terminal. This connector is called a DB-25 or 25-pin D-shell connector, because it's shaped like the letter *D*.

The serial ports on IBM's original PC and XT (8088-class machines) used DB-25 connectors. However, when the IBM PC/AT came out, IBM introduced a smaller, 9-pin connector that handled just the sig-

RS-232C SIGNALS AND PINOUTS



Pin	Description	Abbreviation	Source
1	Protective Ground	GND	
14	Secondary Transmit Data	DTE	
2	Transmitted Data	TD *	DTE
15	Transmit Signal Element Timing (DCE)	DCE	
3	Received Data	RD *	DCE
16	Secondary Received Data	DCE	
4	Request to Send	RTS *	DTE
17	Received Signal Clock (DCE)	DCE	
5	Clear to Send	CTS *	DCE
18	Unassigned		
6	Data Set Ready	DSR *	DCE
19	Secondary Request to Send	DTE	
7	Signal Ground	SG *	
20	Data Terminal Ready	DTR *	DTE
8	Carrier Detect	CD *	DCE
21	Signal Quality Detector	DCE	
9	Reserved (often used for testing)		
22	Ring Indicator	RI *	DCE
10	Reserved (often used for testing)		
23	Data Signal Rate Selector		
11	Unassigned		
24	Transmitter Signal Element Timing	DTE	
12	Secondary Carrier Detect	DCE	
25	Unassigned		
13	Secondary Clear to Send	DCE	

Pinouts of the industry-standard, 25-pin RS-232C connector and the respective functions of all the pins and the directions in which the signals travel. (The acronym DTE stands for *data terminal equipment*—that is, a computer or computer terminal—and indicates that the computer sends the signal. The acronym DCE stands for *data communications equipment* and tells you that the modem sends the signal.) The signals marked by asterisks are present on IBM's 9-pin serial cable.

used by a serial LAN. A SCSI interface also is an option for some peripherals. SCSI, the Small Computer System Interface, offers faster data transmission than a parallel port and may be used not only for printers but for hard drives, CD-ROM drives, scanners, and dozens of other types of peripherals.

Boosting Serial Port Performance

If you're working with a high-speed modem or a direct high-speed communications link to another computer, you may find that your system can't keep up with the flood of characters it's receiving. Moving to a faster machine may help, but another, less expensive approach is worth considering. As I mentioned earlier, most machines that are compatible with the IBM PC/AT use a UART chip called the 16450A. This is a good chip, but it has a potential problem: It can store only one incoming character while it's receiving a second. If two characters come in before the computer has a chance to service the UART, the first character is irretrievably lost.

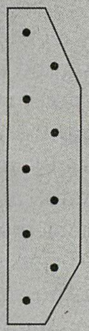
The solution is to get a more advanced UART—the 16550A—which contains a device known as a first-in/first-out (FIFO) buffer. This UART can store up to 16 incoming characters, which allows the system plenty of time to pick them up before they're lost. Upgrading to the new UART is a fairly simple procedure. Because the 16550A can fit into exactly the same socket as the 16450A, you need not replace your serial card or internal modem to do this upgrade; you can simply replace the chip.

If you don't feel comfortable replacing your own UART chip, or if you want one of the best serial port cards available, you can make the upgrade nals that the PC used. This smaller format allowed IBM to fit two serial ports (rather than one) on the back of a single peripheral card.

IBM's 9-pin male connector has become a standard itself; you'll see it gracing the backs of many brands of machines—IBM-compatible and otherwise. But not all manufacturers have adopted it. The serial ports on the Apple Macintosh, for instance, use 9-pin connectors, but they don't follow the arrangement of pins (pinouts) used by IBM and consequently can't use the same modem cables.

—**B.G.**

IBM PC RS-232C SIGNALS AND PINOUTS



Pin	Description	Abbreviation	Source
1	Carrier Detect	CD	DCE
6	Data Set Ready	DSR	DCE
2	Received Data	RD	DCE
7	Request to Send	RTS	DTE
3	Transmitted Data	TD	DTE
8	Clear to Send	CTS	DCE
4	Data Terminal Ready	DTR	DTE
9	Ring Indicator	RI	DCE
5	Signal Ground	SG	

The pinouts for IBM's de facto standard 9-pin serial port connector, introduced in the IBM PC/AT.

by buying a Hayes ESP card. This card contains two serial ports, each with its own 16550A UART, plus a special direct memory-access circuit that can transfer characters to memory automatically. When used in combination with the software that supports it, the ESP card is the highest-performance serial interface available for IBM compatibles.

Serial Ports and Windows

When your system runs under DOS, you can't keep multiple applications from trying to use a serial port at the same time. But, in 386 Enhanced Mode, Microsoft Windows 3.0 and later tries to ensure that no more than one program uses a port at any time. To use this feature, click on the 386 Enhanced icon in the Windows Control Panel. The window that pops up will allow you to change several settings—including one called Device Contention-Ports. For each serial port in your system, Windows lets you select one of three policies for conflict resolution: the Never Warn option, which lets any program use a port at any time; the Always Warn, which asks you to award the port to one of the applications when a conflict arises; and the Idle option, which asks you to award the port to a particular application only if the port hasn't been idle for a certain number of minutes. You can control the status of each port independently.

Another group of Windows settings that's important to serial port users is contained in SYSTEM.INI, one of Windows' configuration files. One setting called COMBoostTime, can give a communications program running in a DOS session extra time to process serial characters. If you find that a communications program is running sluggishly or dropping characters in a DOS session, try setting COMBoostTime to 10 (the default is 2) in the [386Enh] section of SYSTEM.INI.

Also in SYSTEM.INI are four more settings you'll rarely want to change. When Windows starts, it normally makes sure the FIFO buffers on any communications ports with 16550A UARTs are turned on and operating. (They will be if the setting for COM1FIFO through COM4FIFO are set as true.) But if you have a reason for not wanting a FIFO enabled, you can set one of these values to False. (Even though this setting is in the [386Enh] section of the file, it has the same effect in Standard Mode.)

A Port for All Seasons

The original IBM PC's serial ports certainly could have been designed better but, considering the flaws in the way they were implemented, they work remarkably well on today's PC compatibles. While the rules regarding proper configuration of serial ports aren't well-known (even experienced system integrators have been known to violate them), the information presented here should help you achieve reliable serial port operation on any PC-compatible system. ■

Computer consultant Brett Glass is a columnist for Infoworld, *a weekly magazine covering the personal computing industry*

Protect Your Data From Computer Viruses

With the incidence of infection on the rise, now is the time to educate yourself and *take preventive measures.*

by Alan L. Zeichick

Computer viruses. You've probably read about them in *The Wall Street Journal* or *Time*, and your thoughts quite possibly run in one of two directions: a skeptical "What'll those tabloid gossipmongers think of next?" or a panicked "Oh, no! What should I do?" Both responses are justified. True, much of the furor about computer viruses and other such destructive software is hype, but the phenomenon can't be dismissed idly.

According to Philip Chapick, the director of the San Francisco-based Computer Security Institute, "Computer viruses are a serious, but controllable, problem." How bad is the situation? Millions of dollars per year have been lost in productivity and revenue. "The damages tend to be more inconvenience than consequential," Chapnick believes. "The costs come from cleanup and data recovery, not typically from reconstructing permanently lost information."

Just what are computer viruses? Where do they come from, and what you can do about them?

Profile of a Computer Virus

Viruses are computer programs designed to interfere with the proper functioning of your computer. They can operate in several ways, but their most common routine is to "infect" a program by hiding at the back of code so they can't be seen by casual searches. But viruses also can hide in unused parts of your hard disk or floppies, in the boot sector, or even in unused directory spaces.

Viruses are merely an example of one class of "malignant," or "rogue," programs that infects a legitimate program already on your disk. A virus spreads by making copies of itself, usually every time you attempt to execute the legitimate program. Other types of malignant programs include Trojan horses, programs that pretend to be legitimate, but actually are designed to damage your computer. Worms are self-contained programs designed to run on multitasking computers, such as Unix systems or mainframes; they typically are disguised as background processes. Logic bombs are programs altered by an in-house employee so that they do something destructive when a certain condition is met. The dismissal and subsequent removal of the employee's name from the company payroll roster might, for instance, trigger the erasure of vital data. The term *virus* has caught the fancy of the popular press, however, so you'll hear most malignant programs referred to as viruses.

Four main types of viruses can infect DOS-compatible computers, and they're the ones you need to watch for:

- *Program viruses* hide in the .COM (especially COMMAND.COM, DOS's internal command interpreter), .EXE, and .SYS files on your floppy or hard disks. If, for example, you have an infected word processor, the virus will modify the word processor to first execute the virus program and then run the word processing functions. Unless the virus code does something bad, you'll probably never even realize that your program is infected.
- *Boot-sector viruses* reside in hidden space on your disks. The boot sector of each of your disks contains a tiny program smart enough to load in, or bootstrap, DOS. If you boot from a floppy disk containing only data, the boot-sector program knows how to issue a "Non-system disk" error message. Because the boot-sec-

tor program is the very first thing executed on your computer, a virus that has infected this program can take complete control of your system and can hide itself from all but the most intensive search.

- *Partition-table viruses* are similar to boot-sector viruses in that they live in disk space that normally is hidden from the user's view—in this case the program on hard disks that describes how the disk is configured. New virus strains that disguise themselves by modifying the file allocation table—or FAT, the disk's table of contents and a complete index to filenames, sizes, and physical locations—are a variation on this theme.
- *Stealth viruses* employ a variety of sophisticated techniques to stay hidden. They can vary their own internal code, which makes searching for particular binary patterns difficult; they can alter DOS by hiding in your computer's memory and then tricking DOS into believing that the memory doesn't exist (and therefore shouldn't be searched for viruses); or they can create files on your hard disk or floppy, but trick the directory command into believing that the files aren't there.

Computer viruses are programs and, as such, can do anything that a legitimate computer program can do. One thing they all have in common, though, is a need to perpetuate their species: Program viruses will try to infect every executable program they can find, and boot-sector viruses will infect the boot sector of every floppy disk inserted into your system. Some viruses are intentionally designed to infect other computers and possibly do damage; some viruses only propagate themselves. Other than wasting disk space, the latter type of virus causes no real trouble. One such nearly harmless virus is called Friday the 13th because, on that date, the virus displays the message "We hope we haven't inconvenienced you." Another innocuous virus, Music Bug, plays musical notes through a PC's speaker.

Literally hundreds of viruses currently exist. For a detailed description of some of the better-known and more troublesome varieties, see the sidebar "The Most Common Viruses" on page 54.

Origin of the Species

Viruses can come from several sources. They can range from experimental strains that have escaped testing labs to those designed intentionally to do as much damage as possible. Some are targeted at certain computer manufacturers' hardware, which opens up a widespread distribution channel for viruses and infected programs. Your equipment can contract a virus in several ways.

If you download and run executable programs from bulletin board systems (BBSes) or on-line services, it's likely that your computer will become infected. Many reputable bulletin board operators do scan newly uploaded files before making them available for public access, but infected files always can slip through. The uploaders aren't always the ones at fault; if a computer system is infected, so are the programs the uploader shares. Be aware that your equipment cannot become infected by reading e-mail or other data files; it is the process of running infected program code that wakes up the virus.

Some companies have strict rules prohibiting the use of public-domain or shareware code. One reason for this is the potential for viral infection; if your computer

TIP ONE-STEP DIRECTORY

Are you looking for a quick way to remove unnecessary directories from your disks? This usually is a two-step process: Delete any files the directory contains and then use DOS's RD command to remove the directory.

To reduce this to a one-step procedure, use any standard text editor to create the following two-line batch file, which I call ERASEDIR.BAT:

```
ECHO Y | DEL %1
RD %1
```

After you create the batch file, you can issue the following command:

```
ERASEDIR TEST
```

to erase all files in a directory called TEST and subsequently remove that directory from your disk.

When you run ERASEDIR.BAT, DOS replaces the two occurences of the %1 parameter with TEST—or any directory name you specify. It also uses the ECHO command to automatically answer yes to the "Are you sure?" prompt that DOS displays before deleting any files that directory contains.

—J.N.

THE BEST MEDICINE

PC owners have many antiviral software and hardware tools from which to choose. Some of the tools automatically scan your system at bootup, examine your boot sector and partition table, check the sizes of important programs, or search for files hiding in your PC's memory or in unused sections of your disk. Other tools search for viruses by looking for characteristic text strings, binary patterns often found on disks, or symptoms of an infection, such as internal alterations to DOS.

One type of antiviral protection tool works only if it is installed before an infection occurs. Such programs work by creating a "signature file" of all programs and important system files on the disk. This file typically contains a unique number, a so-called checksum, that records the sum of all the characters in the file. This value is recomputed every time the system is scanned for a virus; if the signature doesn't match, the file has been modified, possibly by a virus. Some signature-based systems even

PRODUCT INFORMATION

Centel Federal Systems Inc.
11400 Commerce Park Dr.
Reston, VA 22091
800-843-4850
703-758-7000
PC/Assure ($399) offers virus protection and other security features, including data encryption and password control.

Central Point Software Inc.
15220 N.W. Greenbrier Pkwy.
Suite 200
Beaverton, OR 97006
800-445-4208
503-690-8090
Central Point Anti-Virus ($129) scans disks and memory for new viruses and has a memory-resident utility that watches for infections.

Certus International
6896 W. Snowville Rd.
Brecksville, OH 44141
800-722-8737
216-546-1500
Novi ($129) scans memory, files, and boot sectors for viruses. It includes a memory-resident infection protector and is Windows-compatible.

Commcrypt Inc.
10000 Virginia Manor Rd.
Suite 300
Beltsville, MD 20705
800-334-8338
301-470-2500
Detect Plus ($149) uses cryptographic techniques to report changes in disk content, including files, boot record, and partition tables. The program features file recovery and memory-resident utilities.

ComNetco Inc.
15 E. Main St.
Mendham, NJ 07945
201-543-4060
Virusafe ($99) monitors system input/output (I/O) to prevent viruses from infecting or spreading and includes virus-removal and program-integrity utilities.

Enigma Logic Inc.
2151 Salvio St.
Suite 301
Concord, CA 94520
510-827-5707
SafeWord Virus-Safe ($62) protects against past and future viruses by creating digital signatures of all programs and system files.

IMSI
1938 Fourth St.
San Rafael, CA 94901
800-833-4674
415-454-7101
VirusCURE PLUS ($99.95) scans memory and disks for 900 different viruses and includes utilities for removing viruses and preventing an infection.

International Security Technology Inc.
515 Madison Ave.
Suite 3200
New York, NY 10022
212-288-3101
VIRUS-PRO ($99.95) keeps records of important system files to detect virus infection or pirated software.

IP Technologies
3710 S. Susan
Suite 100
Santa Ana, CA 92704
714-549-4284
Virus Guard ($25) keeps a signature file of all uninfected files and prevents unverified programs from being executed.

have a memory-resident component that tests signatures each time a program is executed.

I recommend that you purchase and use an antivirus program and that you keep up with the latest version of whichever package you choose.

If your computer already is infected, you have several choices. Many commercial antiviral products include tools that identify and eradicate the virus so that your data and programs can often be preserved intact. Try one of those if you feel very comfortable with PCs and with DOS. If not, contact a PC expert or a consultant. If you're not sure about what to do, call the manufacturer of the antivirus tool. That company should have more expertise with malignant software than even your office's in-house PC staff, because actual virus encounters in the workplace are still relatively scarce.

This product box lists the most widely known antiviral tools available for DOS systems. Many of these companies offer several solutions and maintain constant updates on their products to keep up with the flood of new viruses.

—A.L.Z.

McAfee Assoc.
3350 Scott Blvd., Bldg. 14
Santa Clara, CA 95054
408-988-3832
ViruScan ($25) scans files and memory for viruses. The Clean utility ($35) removes viruses. McAfee also offers memory-resident utilities that detect virus infections, network virus detectors, and tools that restore damaged files.

Ontrack Computer Systems Inc.
6321 Bury Dr.
Eden Prairie, MN 55346
800-752-1333
612-937-1107
Dr. Solomon's Anti-Virus Toolkit ($149.95) checks boot sectors, partition tables, and executable programs, searching for alterations by creating a signature file. It also allows floppy disks to be encoded so that unencoded, and therefore suspect, disks may not be accessed.

Parsons Technology
One Parsons Dr.
P.O. Box 100
Hiawatha, IA 52233-0100
800-223-6925
319-395-9626
ViruCide Plus ($69) has signature files that detect and remove viruses and scan for viruses present on disk and in system memory. The Shield program that's included monitors computer operation for signs of virus activity.

PC Guardian
118 Alto St.
San Rafael, CA 94901
800-288-8126
415-459-0190
Virus Prevention Plus ($124.95) detects and repairs boot-sector and partition-table viruses and can maintain a signature file.

RG Software Systems Inc.
6900 E. Camelback
Suite 630
Scottsdale, AZ 85251
602-423-8000
Vi-Spy ($149.95) supports Windows and searches for viruses by using known virus characteristics.

Software Concepts Design
P.O. Box 908
Margaretville, NY 12455
607-326-4422
Flu Shot Plus ($19) is a memory-resident tool that prevents virus infections and creates signatures of important system files and programs.

Symantec Corp.
Computing Group
2500 Broadway
Suite 200
Santa Monica, CA 90404
800-441-7234
310-453-4600
Norton AntiVirus ($129.95) scans disks to intercept viruses and repair damaged files; this software also can be run in memory-resident mode.

Techmar Computer Products Inc.
98-11 Queens Blvd.
Suite 2-C
Rego Park, NY 11374
800-922-0015
718-997-6666
AntiVirusPlus ($99.95) searches for viruses and immunizes the computer against infections; it also supports Windows 3.0.

Trend Micro Devices Inc.
2421 West 205th St.
Suite D-100
Torrance, CA 90501
800-228-5651
310-782-8190
PC-cillin (software and hardware, $139) and the newer PC Rx (just software, $69) create a system checksum, monitor memory activity, and make copies of important system files. Trend Micro also offers a family of virus-prevention tools.

XTree Co.
4330 Santa Fe Rd.
San Luis Obispo, CA 93401
800-634-5545
805-541-0604
VirUSafe ($99) detects and identifies viruses in memory, including those it knows nothing about. Virus-removal utilities, including Scan for hard disks, come with the program.

THE MOST COMMON VIRUSES

According to my sources, the following list includes the most common computer viruses. Numerous versions of some of these viruses exist, however, because clever programmers have been finding them and modifying their code to make the viruses more efficient at hiding, at replicating themselves, or at being destructive.

Cascade, a virus that infects .COM files, was first seen in mid-1987. In its earliest form, it was a Trojan horse that disguised itself as a utility to turn off the NumLock key; it was later transformed into a memory-resident virus. Cascade causes all the characters on the screen to collapse down to the bottom of the display. Another version, Cascade-B, causes random reboots. Variants of this virus are known as 1701 and 1704.

Christmas, a virus that displays a Christmas tree on your monitor screen from December 24 through January 1, also destroys partition tables on April 1.

Dark Avenger, which first appeared in California in 1989, infects .COM and .EXE files and can become memory-resident. After every sixteenth file is damaged, the virus code overwrites a sector on the disk, destroying whatever data was there.

Disk Killer, a boot-sector virus that first appeared in early 1989, spreads itself by copying its code to floppy- or hard-disk sectors, possibly destroying whatever data is present. These sectors are then marked as bad in the FAT. A few weeks after the original infection, Disk Killer encrypts the data on the hard disk. The only real recourse is to reformat the hard disk and reconstruct data from backup sources.

Jerusalem-B, the most prevalent PC virus, infects .COM, .EXE, .SYS, and program overlay files. If it becomes memory-resident on Friday the 13th, it will erase every program that's executed on that day. Jerusalem-B can also cause unusual windows to appear on the screen.

Joshi, also known as the Happy Birthday Joshi or Stealth virus, was discovered in India in mid-1990. It infects floppy-disk boot sectors and hard-disk partition tables. On January 5th, the virus refuses to let the computer work unless the user types "Happy Birthday Joshi."

Michelangelo, discovered in the Netherlands in 1990, infects hard-disk partition tables and floppy-disk boot sectors. If the system is infected, the virus waits until March 6th (the birthday of the Italian painter and architect) and then overwrites the hard drive with random garbage, rendering data recovery impossible.

Music Bug, which first surfaced in 1990, is a memory-resident boot-sector virus. If a computer is infected, a clicking sound or a fragment of music will play just before the boot process is completed. Music can also be heard during disk accesses. Damage is usually restricted to annoyance and lost disk sectors.

Stoned, first found in New Zealand in 1988, has since mutated (with human assistance) into several forms. Most are memory-resident viruses that infect both hard and floppy disks. The virus hides itself by moving the disk's boot sector and putting its own code where the boot sector should be. In addition to displaying rude messages on the screen, Stoned eventually will corrupt your hard disk's FAT.

—A.L.Z.

doesn't come into contact with outside data, it can't pick up an infection. Being cautious, however, can minimize your risks. Because I personally download a lot of code from commercial services, I make a point of using virus-scanning tools to examine new programs before I run them. I then test them on an isolated PC before loading them onto my main computer. To my knowledge, I've never had an infection on any of my systems.

Viruses have been found on shrink-wrapped program disks from software companies, and even on the free utility disks shipped from some clone makers. My advice is to scan every disk the first time you use it, no matter what the source. I've seen disks, fresh from the developer, infected with the Music Bug virus (the vendors fixed the problem as soon as they were made aware of it).

Your equipment also can contract a virus through your local-area network. If an infected program is somehow placed in a public directory, every PC on the network may quickly become infected. I know of one Fortune 500 company, for instance, that had to shut down every PC at one of its satellite facilities for two days while it disinfected hundreds of computers and several network servers. Unfortunately, as a network user you can do everything right, and your computer might still contract an infection. You can help prevent infection, however, by not bringing disks from home to work, or vice versa, without scanning them first.

If you've ever turned on your computer and received a "Non-system disk or disk error" message, you probably tried to boot with a data disk in drive A. You also may have infected your computer.

Remember the boot-sector program mentioned above? Every formatted disk has one, whether that disk is used for programs or data, and your PC tries to execute this program on your data disk. If a floppy disk's boot sector is infected, a failed startup attempt means a virus has infected your hard disk's boot sector. Make certain you check your floppy drives and remove any disks they contain *before* turning on your PC and before rebooting.

Diagnosing an Infection

The best way to find out if your computer is infected by a virus is to purchase and routinely use one of the commercial virus-detection tools listed in the sidebar "The Best Medicine" on page 52. I heartily recommend that you purchase such a tool for yourself; everyone with whom you share should do the same—out of friendship or company spirit. If you don't yet have any of these tools, you can perform the following quick tests to see if your equipment is infected.

At the DOS prompt, type `CHKDSK`. The next-to-last line that appears should read "655,360 total bytes memory" if you have 640K or more of memory installed. If it shows a smaller value, you may have a stealth-type virus in your system; many stealth viruses hide in upper memory and adjust DOS's internal memory value to protect their hiding places. To determine if this is the case, find your original DOS disk, write-protect it (it is vital that you do this, so as not to infect it), and reboot by turning off your computer and turning it on again with the DOS disk in drive A. (Some viruses can trap the Ctrl-Alt-Del reboot sequence and maintain their grip on high memory.) Rerun CHKDSK. Is the total memory now 655,360? If not, your equiment is infected.

If the previous CHKDSK turned up a lot of cross-linked files, you may have a virus that is trying to fool your DIR command, but at the expense of revealing itself to CHKDSK. Do not run CHKDSK with the /F option if you suspect a viral infection. You may destroy your data because CHKDSK will try to "fix" data that the virus tells it to look for.

When viruses infect your program files, they often increase the physical size of the programs to accommodate their own extra program code. Make a note of how many bytes an uninfected copy of COMMAND.COM requires (it varies according to your DOS version, so you'll need to take a directory of your write-protected DOS floppy and compare its size against the copy of COMMAND.COM you think is infected). If the COMMAND.COM on your hard disk suddenly changes size, your equipment is infected.

An Ounce of Prevention

Currently, the chances of your obtaining a computer virus are small, but the number of released viruses is growing every day. At the same time, viruses are becoming more sophisticated. Now is the time to take a few precautions:

- If you use a LAN, find out if your network administrator can provide adequate security to prevent viruses or other malignant code from entering the system and to detect viruses that may sneak through.
- Purchase and install virus-protection software. Make sure that everyone you work with uses some form of virus protection, as well. One caveat: No matter what the software manufacturer's claim, no software-only security system is foolproof. The only way to entirely protect your disk is to write-protect it.
- Know where your disks come from and, even though it's an inconvenience, scan everything. Be careful when dealing with public-domain and shareware programs. Most of them are safe, and many shareware programs are excellent tools you shouldn't be without. But anything downloaded from a phone line should be suspect until proven safe. The same goes, of course, for shrink-wrapped software disks. ■

Alan L. Zeichick is editorial director for Miller Freeman's high-tech magazine group based in San Francisco.

TIP SAFETY FIRST

The ATTRIB command provides an easy way to stop critical files from being changed, or worse yet, accidentally erased. The following two commands, for example, prevent the installation routines used by some programs from making unauthorized modifications to your CONFIG.SYS and AUTOEXEC.BAT files:

```
ATTRIB +R CONFIG.SYS
ATTRIB +R AUTOEXEC.BAT
```

The +R parameter instructs the ATTRIB command to designate the specified file as a read-only file.

Another important file you might consider protecting with the ATTRIB command is DOS's command interpreter, COMMAND.COM. Should you ever need to modify this or any other protected file, you can simply use ATTRIB's -R parameter to list the file's attribute status as read/write.

—J.N.

Programming From A to Z

QBasic comes bundled with every copy of DOS 5.0. You'll find it easy and worthwhile to put this plain-English programming language to work.

by John Wolfskill

Call us old-fashioned, if you like, but the editors of *DOS Resource Guide* and I agree on one point: Basic always has been, and will remain, the ideal programming language for millions of PC users who need to solve everyday problems. For this reason, we've developed a special three-part series to take you from the first steps of learning how a Basic program is written and how it operates through the process of writing your own simple programs.

Why, you might ask, with the thousands of commercial programs available, would anyone turn to Basic or any other a programming language for help? Well, as with snowflakes, no two computing problems are exactly alike. Unfortunately, most commercial PC applications are designed to solve problems that affect a large number of PC users. The truth is, you can't always count on someone else to solve your computing problems for you.

Take Dave, for example. If 10 million software programs were available, they probably still wouldn't help this machinist operate his particular make and model of lathe. Dave wants an interface program that will automate the cutting process on his machine, but none exists. Mary is a schoolteacher whose students speak Swahili. She would like to provide them with an interactive English-Swahili dictionary, but she can't find a program that accommodates this East African language. Harry owns a bakery. Even if he never writes a line of computer code to help run his bakery more efficiently, knowledge of programming theory and techniques will let him better understand how the commercial applications that he uses operate.

Knowing a programming language is a powerful equalizer that can provide each of us the ability to solve our own problems. But how can you become a good programmer without spending years in a college classroom? The very best way I know is to study the work of other programmers.

So, to get you started, I've written a QBasic program called PPRIMER.BAS (the first *P* stands for *printer* and the .BAS extension says you are dealing with a Basic program). It's a practical utility program that lets you customize your printer setup and tap your printer's special features from the DOS prompt (see the Program Listing on page 58 and the sidebar "How PPRIMER.BAS Works" on page 60). I wrote it when I bought a brand-new 24-pin printer and it refused to work with my word processor. Understanding how to program in Basic helped me solve this fairly common—and frustrating—incompatibility.

You can type in the PPRIMER.BAS listing—the source code—and use it immediately, provided you own a copy of the general-purpose programming language

QBasic or Microsoft Quick Basic and have access to a list of your printer's control codes. (For an explanation of why *DOS Resource Guide* chose QBasic for this tutorial, see the sidebar "Why Use QBasic?" below; for an additional primer on using QBasic, see "Elementary Steps to Programming" [*DRG* #4, page 63]).

Later in this installment, after you've had the opportunity to give the program a test run, I'll analyze the listing to show you how and why the program works—and even suggest ways to improve on it. Because others also may need to use the programs you create. In an upcoming installment, I'll also touch on the basics of designing a user interface.

Throughout this series, I'll demonstrate a variety of techniques you can use, show you how to reduce programming to simple concepts, and attempt to remove the hurdles that can make programming a difficult subject to learn. If you decide that Basic programming isn't your cup of tea, you won't have spent your time needlessly. You'll have a utility program you can use to solve at least one of your computing problems. So what have you got to lose?

The Basics

A computer program is a series of coded instructions your PC executes in the order you specify. It is a computer's ability to perform logical operations (calculations, such as addition and subtraction, or comparing two values to determine if they are equal) and its ability to process those results that set it apart from other calculating devices.

In Basic—which stands for Beginner's All-purpose Symbolic Instruction Code—you create instructions for the computer to use by arranging a series of English-like commands or statements (called mnemonics) into a standard text file to create a source listing. Each statement represents an action your computer will take toward a final goal.

After you type in the source listing and start the program, the command interpreter (QBASIC.EXE, in this case) reads the commands in your source listing, then quickly transforms each one into a series of instructions for your system's central processor, the CPU. CPU instructions are written in the native binary language of your computer. When your computer has executed each statement in order, the program is done and, if the program is well-conceived and -written, you've accomplished your goal.

Take a look at the source listing for PPRIMER.BAS on page 58. You'll notice that to the right of each statement is an English-language comment explaining the purpose of the adjacent computer code. Also, note that each major program routine contains a reference letter (I'll talk more about those later) and a comment line that explains what the routine is designed to accomplish. If you're not familiar with a particular command or statement, you might want to look it up in your Basic reference manual before you proceed.

After you think you understand how the command works, type in a small section of the source code listing. The key to understanding a complex task or problem—such as how a computer program operates—is to break it down into small parts. After you understand how each part functions by itself, you'll find it relatively easy to fit the pieces together.

After entering the entire listing, take the program for a trial run to shake out the inevitable typos (not mine, I hope), then go back and look at the source listing a little more closely. This time, try to "think like a computer" as you analyze the program line by line. For example, if a particular section of the program seems to operate in a loop (in other words, a set of statements are executed re-

WHY USE QBASIC?

Programming software actually has three components. The one that creates, edits, and manages text files from the words you input in writing a program is called a text editor. The one that translates the text of a programming language into code the computer can understand is an interpreter or compiler. And the one that helps you correct errors in the logic of the program you've written is a debugger.

QBasic offers other benefits, as well. It is available to everyone who uses DOS 5.0, since QBASIC.EXE comes on each DOS 5.0 system disk. Also, QBasic is compatible with Microsoft's popular Quick Basic, a version of the Basic language that lets you create standalone programs you can execute directly from the DOS prompt.

Unlike early versions of Basic, QBASIC.EXE and Quick Basic have evolved into modern programming languages that help you build sophisticated professional applications. And, they're easy to learn and use. Moreover, these versions of Basic provide the ideal stepping stone for learning other popular programming languages, such as C, Fortran, or Cobol.

—J.W.

peatedly), look for a string of characters—such as B$ in section Q—whose value changes as the program runs.

Although this method may seem a bit awkward at first, you'll find it's a great way to learn programming. And, it's the only practical way to comprehend other people's programs. (You may be surprised to find there are several ways to write a program that produces the same results.) Moreover, this approach will provide you with valuable insights you can use to design your own programs.

Anatomy of a Computer Program

All computers programs consist of a maximum of four types of code. To design a program that works efficiently, you must be able to identify the following functions in the program's task:

- initialization
- input
- process
- output

Initialization Code. The initialization block always is the first element of a computer program. Without an initialization block, your program could not run or would produce wildly improper results. Like the introduction to a play, the initialization block sets the stage for the actions that follow.

During initialization, your program tells the computer how to handle the data it will receive from outside sources. This section also may be used to load data files the program will need on start-up.

If your program requires internal data (information stored within the program, perhaps as a variable that must contain a certain initial value) the variable name and values also should be established during initialization. Sections A to J of the PPRIMER.BAS source-code listing provide an example of the elements a programmer might cover in initialization.

Typically, this portion of the program also prepares the various input and output devices (keyboard, video display, printer) that every program requires for operation, and it tells the computer what to do if a program error occurs.

Below is an example of a program initialization routine. As in PPRINTER.BAS, the Basic code for the computer to execute is on the left, and a remark—a description of the action that should occur—appears after the apostrophe on the right. The second line of the routine initializes variables A through Z by telling QBasic to treat them as integers. The third line prepares the screen's

Program Listing. PPRIMER.BAS, an example program designed to assist you in learning to program in Basic, also has a practical side. After typing it in, you can use it to send control strings to your printer, telling it to do such things as underline, italicize, and boldface text.

```
'______________________________________
' Requirements: Microsoft QuickBasic 4.x, or DOS 5.x QBasic
  Editor
' Date        : 3/1/91
'______________________________________

'(A) - Program initialization -

DEFINT A-Z                      'define all variables as integers

'(B) - Set up error handler -

ON ERROR GOTO TESTERROR         'if program error, goto this label

'(C) - Initialize the screen -

COLOR 7, 1                      'set colors (white on blue)
CLS                             'clear the screen

'(D) - Initialize string arrays and define strings -

DIM ACTION$(6 TO 17), CONTROL$(6 TO 17), MESSAGE$(1 TO 5)
NUM$ = " A B C D E F G H I J K L"
RET$ = "<" + STRING$(2, 196) + CHR$(217)
TE$ = "  to EXIT "

'(E) - Define the user input prompts -

MESSAGE$(1) = " <S>end      <A>ssign      <Q>uit    "
MESSAGE$(2) = "  <A>ction     <C>ontrol     <E>xit    "
MESSAGE$(3) = RET$ + "Send     <A-L> Select    <ESC> Exit"
MESSAGE$(4) = " Type Action String     " + RET$ + TE$
MESSAGE$(5) = "Type Control String     " + RET$ + TE$

'(F) - Define the editing control keys -

UPKEY$ = CHR$(0) + CHR$(72)             'up-arrow key
DOWNKEY$ = CHR$(0) + CHR$(80)           'down-arrow key
LEFTKEY$ = CHR$(0) + CHR$(75)           'left-arrow key
RIGHTKEY$ = CHR$(0) + CHR$(77)          'right-arrow key
HOME$ = CHR$(0) + CHR$(71)              'home key
END$ = CHR$(0) + CHR$(79)               'end key

'(G) - Load the program data file from disk -

GOSUB LOADFILE

'(H) - Display the program screen -

GOSUB INITSCREEN

'(I) - Display the main prompt -
```

➛ Listing continued

→ Listing continued

```
    MESSAGE = 1              'select a message
    GOSUB DISPLAYMESSAGE     'and display it
    COLOR 7, 0               'set screen colors (white on black)

   '(J) - Display the data items -

    GOSUB DISPLAYDATA

   '(K) - Main processing loop -

    DO                       'top of main processing loop
    X$ = INKEY$              'check for a keystroke
     X$ = UCASE$(X$)         'force ASCII value to uppercase
      SELECT CASE X$         'define key case structure:

       CASE "S"          'if the S key was pressed
        MAXPOS = 26      'assign maximum cursor column position
        MINPOS = 5       'assign minimum cursor column position
        GOSUB SEND       'send a control string to the printer
        COLOR 7, 0       'set colors (white on black)
        GOSUB GATHER     'read the screen, print the message
        MESSAGE = 1          'select a message
        GOSUB DISPLAYMESSAGE 'and display it

       CASE "A"              'if the A key was pressed
        GOSUB SELECTFUNCTION 'select a program editing function
        IF CANCELKEY THEN    'if the cancel flag is Select
         CANCELKEY = 0       'reset it
         GOSUB DISPLAYDATA   'redisplay the data items
        END IF
                             'continue with this loop
       CASE "Q"              'if the Q key was pressed
        GOSUB SAVEFILE       'save the current data
        CLS                  'clear the screen
        END                  'exit the program to DOS or QBasic

       CASE ELSE
        END SELECT           'end of key case structure
   LOOP                      'bottom of main processing loop:

   '(L) - Select function routine -

  SELECTFUNCTION:

    MESSAGE = 2              'select a message
    GOSUB DISPLAYMESSAGE     'and display it

   DO                        'top of select function loop:
    X$ = INKEY$              'check the keyboard for a
  keystroke
      X$ = UCASE$(X$)        'force the ASCII value to
  uppercase
        SELECT CASE (X$)     'define a key case structure:

         CASE "A"            'if the A key was pressed
 MESSAGE = 4           'select a message
```

Listing continued on p. 75

foreground and background colors, and the final line clears the screen to the chosen settings.

```
'- Initialization -
DEFINT A-Z  ' Define all variables as integers
COLOR 15,1  ' Set screen colors (white on blue)
CLS         ' Clear the screen
```

Input Code. For your computer program to accomplish anything practical, it requires input from one or more external devices. These include the keyboard, disk drives, serial communications ports, or a mouse pointer. The following is an example of Basic program statements that gather information from the keyboard:

```
' - Input Code -
INPUT A  ' ? prompt cues the user to type a number
INPUT B  ' ? prompt cues the user to type another
  number
```

Input code isn't restricted to INPUT statements that prompt the user to type in a response. Statements such as:

```
X$ = INKEY$
```

and:

```
X$ = UCASE$(X$)
```

in section K also gather input.

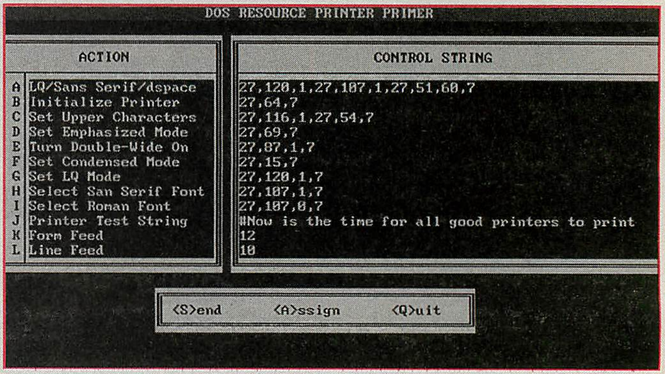
Process Code. Every program you write also must contain at least a few lines of code that process information, first defining and then solving a problem.

For example, a typical process may begin by asking the user to supply two numbers. After the user enters the requested information, the process stores each value in a program variable, and adds the values represented by the variables to produce a result. The commands for completing this simple process might look like the following:

```
'- Process Code -
C = A + B
```

One of the most important parts of designing your program is identifing the problems that must be solved, and turning those problems into lines of process code. The CASE statements in section K are examples of process code at work. They do one thing if the key pressed was an *A* and something else if an *S* was pressed.

HOW PPRIMER.BAS WORKS



```
                        DOS RESOURCE PRINTER PRIMER

        ACTION                            CONTROL STRING
A LQ/Sans Serif/dspace       27,120,1,27,107,1,27,51,60,7
B Initialize Printer         27,64,7
C Set Upper Characters       27,116,1,27,54,7
D Set Emphasized Mode        27,69,7
E Turn Double-Wide On        27,87,1,7
F Set Condensed Mode         27,15,7
G Set LQ Mode                27,120,1,7
H Select San Serif Font      27,107,1,7
I Select Roman Font          27,107,0,7
J Printer Test String        #Now is the time for all good printers to print
K Form Feed                  12
L Line Feed                  10

              <S>end        <A>ssign        <Q>uit
```

PPRIMER.BAS is a utility that helps you tap most of your printer's special features while you work at the DOS prompt. It lets you send as many as twelve 49-character control strings to any printer; plus, it provides a way to quickly set up your printer before you load software that doesn't understand printer control codes.

The program automatically saves your favorite printer setup strings in a data file. Using PPRIMER.BAS, you can quickly initialize your printer, and change print modes, type styles, margin settings, and more.

In case you're not familiar with printer control codes, they tell your printer to do special things, such as underline or print emphasized, boldfaced, or compressed characters. Although various printer makes and models respond to different codes, most follow the same general rules. The computer sends out a character that tells the printer to regard the characters that follow it as control codes instead of printed characters. Next come one or more characters that define, or delimit, the special print function. A typical control-code string might look like this:

```
27, 64
```

Your printer recognizes the character string as a special code and shifts its printing gears accordingly.

After you've typed in the program, save your work to disk. If you have Quick Basic, you can compile the source listing to create PP.EXE, a program you can run from the DOS prompt. Press Alt-R to bring up Quick Basic's Run menu. Select the MAKE.EXE and Exit option to create a standalone program.

If you're using the Microsoft QBasic interpreter, load the source code by typing:

```
QBASIC PP.BAS
```

Setting Up Shop

You can start the program by pressing the F5 key, but before you do, turn on your parallel printer and place it on-line. The program will display a form, as shown in the Photo. Since PPRIMER.BAS is designed to control any printer, your form initially will be blank. To fill in the blanks with the proper control strings for your own printer, turn to the section of your printer's manual that describes its printing and formatting commands.

Because Epson America Inc. manufactures a popular line of printers, I'll use Epson printer control codes to show you how to use the program. If you own a different printer, your control codes may be similar, but not identical, to mine (see the table "Common Printer Control Strings" at right for a list of common sequences for Epson and HP Laserjet printers).

As a writer, I often deal with editors who demand manuscripts that are double-spaced and easy to read. When that situation arises, I use PPRIMER.BAS for three separate operations:

- setting my Epson LQ-500 to letter-quality mode
- shifting to a bold sans-serif typeface
- sending control codes to adjust the line spacing to achieve double spacing

With PPRIMER.BAS, I can do this from the DOS prompt, without resetting my printer's DIP switches or loading my word processor to add extra line feeds to my single-spaced ASCII documents.

My Epson printer's basic command syntax to the set the print mode is:

ESC *x n*

where *ESC* (ASCII 27) is an identifier that tells the printer a control string follows, the lowercase letter *x* (ASCII 120) tells the printer the command is for setting the print mode, and *n* is a print-mode value that selects either draft (n=0) or letter quality (n=1) printing. To tell my printer to get ready to print in letter-quality mode, the correct string is:

```
27 120 1
```

(Note: The value for *n* must be either a zero or 1; do not use the ASCII equivalent of the 1 key. The other control codes, however, do require ASCII values.)

I prefer my printer's bolder-looking sans-serif font to its default type style, so the next step is shifting the printer to this font. On my Epson, the proper syntax to do this is:

ESC *k n*

where *ESC* (ASCII 27) is the control string identifier, *k* (ASCII 107) is the type style family code, and *n* is the value for the sans-serif font, which is 1. This makes the control string to change fonts:

```
27 107 1
```

To make my printer provide double spacing, I adjust the line spacing in n/180-inch increments using the following syntax:

```
ESC 3 n
```

As before, *ESC* (ASCII 27) is the control string identifier, the 3 key (ASCII 51) selects incremental line spacing, and *n* is a number between zero and 255 that determines the number of n/180-inch steps between each printed line. I use a value of 60 for my documents. This final string reads:

```
27 51 60
```

The completed control string for preparing my manuscripts for editing involves all three actions:

```
27 120 1 27 107 1 27 51 60
```

To enter the string into PPRIMER.BAS, I press A to select the Assign option and press A again to select the Action option. When the flashing cursor appears, I type in a shorthand description I can remember easily, such as:

```
LQ/Sans Serif/dbl space
```

Then I press Enter to quit, press A, and select C to choose the Control option. When the flashing cursor appears, I type the control string on the line adjacent to my description, separating control characters with a comma:

```
27,120,1,27,107,1,27,51,60,7
```

Notice that I've added the digit 7 to the end of the string. This value tells Epson printers to print the BEL character, which makes the computer sound a short electronic beep to let you know it has received the control string. This isn't necessary, but I find it handy. If your printer doesn't support the BEL code, omit the last comma and character code.

Press Enter to quit. Then press S and Enter to select this choice and send it to your printer. As you begin to accumulate different control strings, you'll need to press the up- or down-arrow key to highlight your selection (or press the corresponding letter to move the cursor to this selection).

Testing 1,2,3

To ensure that your control codes are working as they should, PPRIMER.BAS lets you enter a text string to test your setup. Text strings must begin with the ASCII pound sign (#) so the program will recognize them as text. Unlike control strings, the words in your text strings may not contain commas.

To use the text string to test a printer-control string, first send the control string to your printer. Then, send the text string. PPRIMER.BAS strips off the initial pound sign and sends the text to the printer. Check to ensure the text looks the way you'd like.

Of course, not everyone has an Epson printer, so I have included printer control sequences for another popular type of printer, the Hewlett-Packard LaserJet II series.

—J.W.

COMMON PRINTER CONTROL STRINGS

EPSON FX/LQ SERIES PRINTERS

Action	Control String
Initialize printer	27,64,7
Select roman (upright) type	27,107,0,7
Select italic type	27,52,7
Select boldface type	27,87,1,7
Turn underline on/off	27,45,7
Select condensed	27,15,7
Select graphics mode	27,42,7
Select letter-quality mode	27,120,1,7
Send form feed	12
Send line feed	10
Select uppercase	27,116,1,27,54,7
Select subscript	27,83,7
Select letter-quality justification	27,97,1

HEWLETT-PACKARD LASERJET II SERIES PRINTERS

Action	Control String
Initialize printer	27,69,7
Eject page	27,38,124,48,72,7
Send form feed	27,38,124,50,72,7
Select portrait mode	27,38,124,48,79,7
Select landscape mode	27,38,124,49,79,7
Select 10 characters per inch	27,40,115,49,48,72,7
Select 12 characters per inch	27,40,115,49,50,72,7
Select proportional spacing	27,41,115,49,80,7
Select fixed spacing	27,41,115,48,80,7
Select 8-point type	27,40,115,56,86,7
Select 12-point type	27,40,115,49,50,86,7
Select roman (upright) type	27,41,115,48,83,7
Select italic type	27,41,115,49,83,7

We're Looking for a Few Good Tips!

Have you ever discovered a new way of using a DOS command or written a clever batch file to perform a repetitive task? Sure you have! And *DOS Resource Guide* wants to publish your secrets. Send us your best tips and hints for publication and receive the undying gratitude of DOS users around the world.

Send your best DOS tips and batch files to:

DOS Resource Guide
80 Elm Street
Peterborough, NH 03458

Or drop us a line on-line:

Compuserve (75300, 2361)
MCI Mail (469-7260)

Output Code. The final step in designing a practical computer program is returning results to the user. The output section contains commands or statements that tell the computer where to display, or print the results. Some examples of output devices are the video display, disk drives, and your printer.

The code for a simple output block may look like this:

```
'– Output Code –
CLS            ' Clear the display
LOCATE 12,20   ' Put cursor at line 12, position 20
PRINT C;       ' Print the result of the process
```

Section S, which controls the appearance of the PPRIMER.BAS screen, provides a good example of output code.

If we place all of the above examples together, we have a small, but functional computer program:

```
'– Initialization –
DEFINT A-Z      ' Define all program variables as integers
COLOR 15,1      ' Set screen colors (white on blue)
CLS             ' Clear the screen
'– Input –
INPUT A         ' Ask the user to type a value
INPUT B         ' Ask the user to type another
'– Process –
C = A + B       ' Process the input. Create a result
'– Output –
CLS             ' Clear the display
LOCATE 12,20    ' Locate the cursor at line 12, position 20
PRINT C;        ' Print the result
'– End the program –
END
```

This program sets up the screen to display white text on a blue background, asks the user to type two values, adds those values, clears the screen, positions the cursor and prints the sum of the two values.

Stay Tuned

In the next issue, I'll break apart PPRIMER.BAS's source listing, explaining program commands and statements, instructions, and how to define your problem. I'll also discuss loops, subroutines, CASE structures, and much more.

In the meantime, check all the Basic source listings you can find. In each listing, try to identify the functional program blocks we've discussed. You may even want to try designing a few small programs of your own. ■

John Wolfskill is a free-lance writer and president of Powertrain Development, a New Hampshire-based software development company.

Box It Up

With a little help from these short batch files, you can turn your keyboard into a designer's tool.

by Neil C. Burnett

Sometimes the simplest tools are best. You can create fancy-looking boxes within reports and letters using a desktop publishing package, but often it takes a while to master the ins and outs of programs like these. My short batch files let you create boxes edged with a single or double rule—in a jiffy—and they won't cost you a penny.

You'll find a dozen uses for the boxes that SBOX.BAT and DBOX.BAT can produce (see Program Listings 1 and 2). You might box up a report title or create a double line down the side of a page to indicate an area for notes on the printed version of a report. You could use the programs to add boxes to charts and tables or, more practically, to create attractive on-screen menus for your programs.

The batch files below alter a key's ASCII assignment. The changes require that DOS's ANSI.SYS program (keeper of the ASCII table) be present in memory. To put it there, add a line such as the following to your CONFIG.SYS file:

```
DEVICE=C:\DOS\ANSI.SYS
```

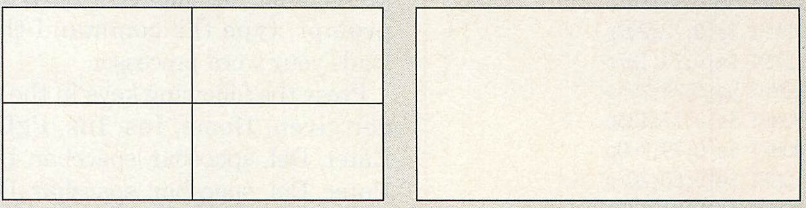
and reboot your computer to have this change take effect. (This example assumes that ANSI.SYS resides in a subdirectory named \DOS.)

SBOX.BAT and DBOX.BAT redefine the following keys: Ins, Del, End, down arrow, PgDn, left arrow, right arrow, Home, up arrow, PgUp, and asterisk. The change affects these keys no matter where they are positioned on the keyboard. If you have an extended keyboard, for example, both the keys on the numeric keypad and the keys in the two key clusters to the left of the keypad will change. Of course, if you press the NumLock key, you can type numbers on the keypad and use the dedicated cursor and screen-movement keys to draw boxes.

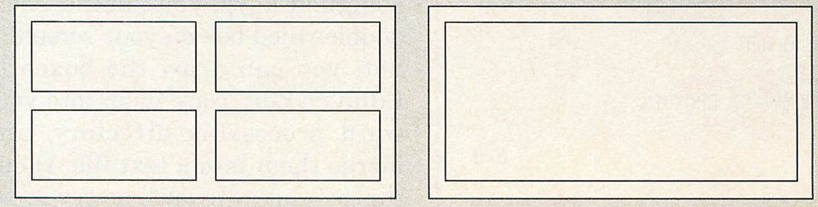
SBOX.BAT lets you create boxes with a single rule; DBOX.BAT creates boxes with a double rule. (The final PROMPT line in each program alters the DOS prompt

SAMPLE BOXES

Boxes created with SBOX.BAT



Boxes created with DBOX.BAT



Program Listing 1. SBOX.BAT lets you create single-ruled boxes.

```
CLS
PROMPT $e[0;71;218p
PROMPT $e[0;72;194p
PROMPT $e[0;73;191p
PROMPT $e[0;75;195p
PROMPT $e[0;77;180p
PROMPT $e[0;79;192p
PROMPT $e[0;80;193p
PROMPT $e[0;81;217p
PROMPT $e[0;82;196p
PROMPT $e[0;83;179p
PROMPT $e[42;197p
ECHO OFF
CLS
PROMPT $P SBOX $G
```

End

so you know which of the programs you're working in.) You may make the boxes as large as you want by repeating keystrokes and using spaces to fill the center of the box. You also may add horizontal or vertical lines to subdivide a box as shown in the figures in "Sample Boxes." But each line of your box must end with a carriage return.

Program Listing 2. DBOX.BAT let you create double-ruled boxes.

```
CLS
PROMPT $e[0;71;201p
PROMPT $e[0;72;203p
PROMPT $e[0;73;187p
PROMPT $e[0;75;204p
PROMPT $e[0;77;185p
PROMPT $e[0;79;200p
PROMPT $e[0;80;202p
PROMPT $e[0;81;188p
PROMPT $e[0;82;205p
PROMPT $e[0;83;186p
PROMPT $e[42;206p
ECHO OFF
CLS
PROMPT $P DBOX $G
```

End

Creating the Listings

To use SBOX.BAT and DBOX.BAT, you must type each one into a text file using DOS's Edlin line editor, Version 5.0's Edit full-screen editor, or a word processor that saves text in ASCII format. Copy them carefully, paying special attention to spaces and punctuation. I've used a colon (:) in lieu of REM (remark) statements. The colon achieves the same end: The information to the right of the colon is ignored by the computer. I prefer colons because I think they produce a cleaner looking program.

To run DBOX.BAT, for instance, make certain that ANSI.SYS is loaded into memory and invoke DBOX.BAT at the DOS prompt. Then, press Enter. You should see a C:\ DBOX > prompt. If you don't, you've made a typing error and must find it before you can go any further.

If the prompt is correct, press the following keys as a quick test: Home, Ins, Ins, and PgUp. This should create the upper portion of a double-ruled box. The table, "Key Assignment Changes," shows you which key to press to produce a certain box-drawing character.

The logical place to employ these boxes is within your word processor. Unfortunately, some word processors—Microsoft Works, among them—don't let you reassign the keys listed above. To test your word processor, follow the steps outlined above but, at the C:\ DBOX > prompt, type the command that loads your word processor.

Press the following keys in the order given: Home, Ins, Ins, PgUp, Enter, Del, spacebar, spacebar, Del, Enter, Del, spacebar, spacebar, Del, Enter, End, Ins, Ins, and PgDn. If your word processor accepts the reassigned keys, you should see a double-ruled box on your screen. If not, you can draw the boxes in Edlin or Edit, copy them into your word processing directory, and merge them into a text file. Doing this is a bit awkward, however, because you must predefine your boxes or rules to fit your text.

When you have finished making boxes and want to restore the original key assignments, run NOBOX.BAT (see Program Listing 3). Your normal DOS prompt should appear. ■

Free-lance writer Neil C. Burnett resides in Titusville, Fla.

Program Listing 3. After you finish drawing boxes wtih SBOX.BAT and DBOX.BAT, use NOBOX.BAT to restore your keyboard's original key assignments.

```
CLS
PROMPT $e[0;71;0;71p
PROMPT $e[0;72;0;72p
PROMPT $e[0;73;0;73p
PROMPT $e[0;75;0;75p
PROMPT $e[0;77;0;77p
PROMPT $e[0;79;0;79p
PROMPT $e[0;80;0;80p
PROMPT $e[0;81;0;81p
PROMPT $e[0;82;0;82p
PROMPT $e[0;83;0;83p
PROMPT $e[42;42p
ECHO OFF
CLS
PROMPT $P $G
```

End

KEY ASSIGNMENT CHANGES

Key	Key Function	ASCII Character
	SBOX.BAT	DBOX.BAT
Del	┌	╔
End	└	╚
Down Arrow	┴	╩
PgDn	┘	╝
Left Arrow	├	╠
Right Arrow	┤	╣
Home	┌	╔
Up Arrow	┬	╦
PgUp	┐	╗
*	┼	╬
Ins	─	═

HARD DISK HOUSEKEEPING

If you've been putting off clearing extraneous files from your hard drive, these batch files may provide the incentive you need.

by Stephen Nebel

Despite our best intentions, they accumulate: .OBJ files left over from sessions with compilers and assemblers, .BAK files from your word processor or occasional encounters with Edlin, .RFT files from word processing conversions, and .$?? files from who knows where—useless files occupying valuable disk real estate.

I've developed a pair of batch files that provide an efficient means of keeping my hard drives free of such deadwood. You can use my technique if you have DOS 5.0, are willing to teach old dogs DEBUG.COM and EDLIN.COM some new tricks, and don't mind traveling a few of DOS's back roads.

Meet the Batch Files

The first batch file is CLEANALL.BAT, a simple program that can repeatedly call the second batch file, CLEANDRV.BAT, each time handing it a different drive-letter parameter. (See Program Listings 1 and 2.) Of course, if your system has only one hard drive, you will need to specify only one drive parameter, and CLEANALL.BAT will need to call CLEANDRV.BAT only once. CLEANDRV.BAT is the workhorse of the pair; it methodically sweeps every subdirectory on the targeted drive clean of unwanted files.

CLEANALL.BAT is very simple. It contains one CALL CLEANDRV line for each hard drive you want to rid of unwanted files. In my computer, there are four hard drives: C, D, E, and F. So, my version of CLEANALL.BAT has four lines calling CLEANDRV, one for each hard drive in my system. You may need more or fewer such lines in your version of CLEANALL.BAT. For instance, if you have only one hard drive, C, on your system, you will remove lines 2, 3, and 4 from CLEANALL.BAT. On the other hand, if you have hard drives C, D, E, F, and G, you will insert an additional line after line 4 to call CLEANDRV for drive G.

Before I explain the workings of the second batch file, CLEANDRV.BAT, in greater detail, let me set up a situation. Suppose you have a hard disk (drive C) with a root directory and two subdirectories: C:\DOS and C:\MASM. Let's also assume you just used Edlin to do some editing on the AUTOEXEC.BAT file in your root directory and have just assembled the program BLASTER.ASM with Microsoft's assembler, MASM, and stored it in C:\MASM. As a result, you now have a file called AUTOEXEC.BAK (Edlin creates a .BAK file every time you edit a file) in your root directory and a file named BLASTER.OBJ in C:\MASM (MASM creates a .OBJ file when it assembles a program). It's unnecessary files like these that you should remove from your hard drive.

Finally, assume that DEBUG.COM and EDLIN.COM are in C:\DOS and that you have included the DOS subdirectory in your PATH command or in the PATH statement of your AUTOEXEC.BAT file. The DOS 5.0 upgrade procedure should have placed EDLIN.COM and DEBUG.COM in your C:\DOS directory. CLEANDRV.BAT and CLEANALL.BAT should be stored in C:\DOS, as well.

To delete the two files mentioned above, you might use the following two-line batch file, even though it would be just as easy to type the two lines at the DOS prompt:

```
DEL C:\AUTOEXEC.BAK
DEL C:\MASM\BLASTER.OBJ
```

Unfortunately, that batch file would work only for the two files mentioned. That's hardly helpful, because most hard disks are in a continual state of flux, with new subdirectories being created and obsolete ones being deleted.

You really need a batch file that figures out the current disk structure automatically and uses that information to delete unwanted files. CLEANALL.BAT and CLEANDRV.BAT do exactly that, with a little assistance from Debug and Edlin.

Inside CLEANDRV.BAT

The purpose of CLEANDRV.BAT is to delete the desired files by creating and running a new batch file, DELELIST.BAT. The latter simply contains a number of DOS DEL (delete) commands that erase the files you no longer want. Using one batch file to create another can be rather complicated, and so it is in this case. CLEANDRV.BAT must employ special features of both Edlin and Debug to create the correct DEL commands and save them in DELELIST.BAT. The other batch file, CLEANALL.BAT, simply calls CLEANDRV.BAT once for each disk drive you want to clean of unwanted files. Refer to the table "CLEANDRV.BAT at Work" for an overview of how the batch file works.

I'll begin discussion of CLEANDRV.BAT at line 4, with the DOS DIR command and DOS redirection. (When you type in the program listings, *do not* type in the line numbers. They are included here for reference only.) If you invoke CLEANDRV.BAT with the command:

```
CLEANDRV C
```

the target drive—the value of the %1 parameter—is C. That means that as DOS processes the batch file, it will substitute a *C* for every occurrence of %1.

TIP KEEPING CURRENT

As a rule, the DOS shell takes responsibility for updating its display to reflect the most current information about your disks and file listings. Every so often, however, DOSSHELL misses something. This generally happens when you engage in an activity outside the shell—perhaps using DOS 5.0's UNDELETE command to recover an accidentally erased file. Should something like this occur, you can use the Refresh option in the pull-down View menu to reread all directory information for the active disk and update your DOS shell display.

—J.N.

Line 4 uses two DIR switches introduced in DOS 5.0: /S and /B. /S tells the system to display a directory not only for the current drive, but for every directory below the current directory. /B further instructs the system to display the full directory path of each file that matches the search criteria. When DOS evaluates line 4, it first will create a complete directory listing of every .BAK file in or below the root directory. The redirection symbol (>) tells DOS to send the directory listing to a file called DELELIST.BAT (instead of sending this information to the screen).

Now comes the start of the automation sequence, the part of the batch file that uses the double redirection symbol (>>) to add information to DELELIST.BAT. If you look at lines 5, 6, and 7, you'll notice that I use a DIR command for each type of file I want to delete. Likewise, in your version of CLEANDRV.BAT you will need one DIR line in this section for each type of file you want to delete. For instance, if the only files you want to delete are .BAK files, you will remove lines 5, 6, and 7 from CLEANDRV.BAT. If you want to delete .BAK and .TMP files, then remove lines 6 and 7, and change line 5 to read:

```
DIR %1:\*.TMP /S /B >> DELELIST.BAT
```

Be sure that the first DIR line you place in this section uses the single redirection symbol, a greater-than symbol (>), while all subsequent lines use the double redirection symbol (>>).

In my example, the batch file finds only two files on my drive C that match the file types specified in lines 4 through 7. They are the previously mentioned AUTOEXEC.BAT and BLASTER.OBJ. So, after DOS finishes executing line 7 of CLEANDRV.BAT, DELELIST.BAT will contain two lines:

```
C:\AUTOEXEC.BAK
C:\MASM\BLASTER.OBJ
```

You can convert these lines into an executable batch file simply by adding a DEL command to the beginning of each line of DELELIST.BAT. The remainder of CLEANDRV.BAT is devoted to doing just that, but it isn't as simple as it sounds.

The Revenge of Edlin

The natural tool for performing this type of insertion, at least in the current context, is Edlin, the text editor that has been distributed with MS-DOS since Version

1.25. It has been much maligned over the years, not without good reason in many cases, but it does have one highly useful property: It can accept commands from a file. That is, you can create a file containing Edlin commands and then use redirection to make Edlin read this so-called script file and carry out the commands. The procedure is the reverse of the one I used earlier to redirect output to a file. Here, I employ redirection to use the contents of a file as input to a program, Edlin.

The bulk of the code in CLEANDRV.BAT—everything from line 9 to line 34—is dedicated to creating a script file that contains commands instructing Edlin to insert the DEL command at the beginning of each line of DELELIST.BAT. CLEANDRV.BAT assembles the script file, EDLINCMD.ASC, segment by segment, using ECHO and COPY commands. The script file eventually is used in line 35 of CLEANDRV.BAT.

The Role of Debug

Unfortunately, the first segment needed for EDLINCMD.ASC cannot be created with ECHO and COPY alone. This segment, EDLINCM1.ASC, first must be created with the ECHO command in line 9, then modified using Debug, another program that has been included with DOS since the early days. The modification consists of trimming the length of EDLINCM1.ASC to 10 bytes. This is necessary because the ECHO command in line 9 that creates EDLINCM1.ASC adds a carriage return (<CR>), a linefeed (<LF>), and an end of file (<EOF>) character to the end of the line it creates. Thus, after line 9 is executed, the file EDLINCM1.ASC contains the single line:

```
1,64000RC:<CR><LF><EOF>
```

TIP QUICK COMMAND

Most computerists have heard about DOSKEY, a new DOS 5.0 utility that lets you recall previously entered commands. Few people, however, realize that DOS provides the following function key shortcuts for quickly recalling those commands after you've loaded DOSKEY into memory:

- F7, which displays a complete list of commands stored in the DOSKEY buffer
- F8, which lets you find commands stored in the DOSKEY buffer that begin with a specified letter
- F9, which displays a "Line number:" prompt. To execute a particular command, you simply press F7 to review the commands and their numbers. Press F9 and type the number of the command you want to execute.

—J.N.

Program Listing 1. CLEANALL.BAT is a simple batch file that can repeatedly call CLEANDRV.BAT. You should include one CALL CLEANDRV line in this batch file for each hard drive you want to rid of unwanted files. If you have only drive C on your system, for instance, remove lines 2, 3, and 4 from this listing. (The line numbers are included as reference points; do not type them when you enter the listing into your text editor or word processor.)

```
1 CALL CLEANDRV C
2 CALL CLEANDRV D
3 CALL CLEANDRV E
4 CALL CLEANDRV F
5 REM —- Go to root directory —-
6 C:
7 CD \
8 DIR
```

End

(The drive letter C is supplied by the first batch file, CLEANALL.BAT, and replaces the parameter %1 the first time CLEANDRV.BAT is called. If, as in Listing 1, additional drives are specified, those letters will replace %1 in sequence each time CLEANDRV.BAT is called by CLEANALL.BAT.)

You cannot stop the ECHO command from adding these three extra characters. The problem is that Edlin needs only the first 10 characters—1,64000RC:—so the <CR>, <LF>, and <EOF> characters must be eliminated before this line is inserted into EDLINCMD.ASC, the Edlin script file. Debug is used to trim the length of the line to 10 characters, thereby eliminating the unwanted characters.

For this task, you must create a second script file—DEBUGCMD.ASC. It contains the instructions that tell Debug how to modify EDLINCM1.ASC. Thus, CLEANDRV.BAT actually involves two script files. One, EDLINCMD.ASC, is needed to tell Edlin how to properly insert the DEL commands into DELELIST.BAT, the batch file that eventually will be called to delete the unwanted files from the hard drive. The other, DEBUGCMD.ASC, tells Debug how to modify EDLINCM1.ASC, one of the segments that makes up EDLINCMD.ASC.

The Debug script file, DEBUGCMD.ASC, is created and assembled—segment by segment—in lines 15 through 25. Let's look at it more closely. With Debug, truncating file is a straightforward operation. The file you need to trim is EDLINCM1.ASC. To do it manually, you type:

```
DEBUG EDLINCM1.ASC
```

and press Enter. At Debug's hyphen prompt, type:

```
RCX
```

and press Enter. Debug will respond by displaying the file length of EDLINCM1.ASC in the CX register, which you can edit. Debug requires you to provide numbers in hexadecimal format, and since you want the length of this file trimmed to 10 characters, you type the hexadecimal equivalent of 10, which is the letter *A*, then press Enter. Press *W* to write the bytes back to the disk, and press *Q* to quit.

That's the manual method. But we want to automate this part of the task, so a simple script file is needed that contains the same keystrokes. The file needs to look like this:

```
RCX
A
W
Q
```

In CLEANDRV.BAT, this file is created using the DOS ECHO command, followed by the DOS COPY command in "concatenation" mode. Lines 16–19 echo each of the command lines to the individual files DEBUGCM1.ASC, DEBUGCM2.ASC, DEBUGCM3.ASC, and DEBUGCM4.ASC. Then, lines 22–25 assemble the four files into the single file DEBUGCMD.ASC. If a previous version of the script file resides on your hard drive, it is erased by line 21.

In lines 22–25, the /B switch puts the COPY command into binary mode, which prevents superfluous end-of-file characters from being appended at the end of individual copy operations. The plus sign (+) included in all COPY commands after the first one tells the system to append a file to the existing file. This process of combining several files into one is called concatenation. (Using the + with the COPY command accomplishes much the same thing that the double redirection symbol (>>) accomplishes with the ECHO command. Both add to an existing file rather than overwriting it with new information.)

Once DOS has executed line 25 of CLEANDRV.BAT, the Debug script file DEBUGCMD.ASC is complete.

As noted earlier, the instructions in this file tell Debug to trim EDLINCM1.ASC to 10 bytes. This occurs in line 27. In plain English, line 27 says, "Run Debug, acting on the file EDLINCM1.ASC using the instructions contained in the file DEBUGCMD.ASC."

Final Steps

After DOS executes line 27, EDLINCM1.ASC is in its proper form and can be combined with the three other segments created in lines 10–14 to make EDLINCMD.ASC. This is accomplished with the COPY command in lines 30–33. At this point, EDLINCMD.ASC finally contains all the instructions Edlin needs to insert the needed DEL commands into DELELIST.BAT. In case you've forgotten, the goal of this entire exercise is to make a new batch file, DELELIST.BAT, that will delete the unwanted files on the hard drive.

CLEANDRV.BAT AT WORK

Lines 4–7	Handle partial creation of DELELIST.BAT, containing the list of files to be deleted but without the necessary DEL command at the beginning of each line.
Lines 9–14	Create the four segments of the Edlin script file, EDLINCMD.ASC. The first segment, EDLINCM1.ASC, must be modified using Debug before it can be used.
Lines 16–20	Create the four segments of the Debug script file, DEBUGCMD.ASC.
Line 21	Deletes the previous version of DEBUGCMD.ASC, if it exists.
Lines 22–25	Combine the four segments of the Debug script file to make DEBUGCMD.ASC.
Line 27	Uses Debug and its script file to modify EDLINCM1.ASC.
Line 29	Deletes the previous version of EDLINCMD.ASC, if it exists.
Lines 30–33	Combine the four segments of the Edlin script file to make EDLINCMD.ASC.
Line 35	Uses Edlin and its script file to modify DELELIST.BAT, inserting a DEL command at the beginning of each line.
Lines 36–37	Delete the unnecessary .ASC files.
Lines 38–45	List the deletions to be performed and give the user an opportunity to abort the procedure.
Line 46	Calls DELELIST.BAT to perform the deletions.
Line 47	Deletes DELELIST.BAT and DELELIST.BAK from the hard drive.

Line 35 invokes Edlin with DELELIST.BAT as the target and EDLINCMD.ASC as the script file, neatly inserting a DEL at the beginning of every line in DELELIST.BAT. This line can be interpreted as "Run Edlin, editing the file DELELIST.BAT using the instructions contained in the file EDLINCMD.ASC." This line inserts a DEL command at the beginning of each line of DELELIST.BAT, the file that was created in lines 4–7. With the work of the many .ASC files complete, they are erased in lines 36 and 37.

DELELIST.BAT now is complete, and if you call it, it will delete the types of files specified in lines 4–7. Before that happens, however, lines 38–44 display a list of the files to be deleted and give you one last chance to abort the process. If you press Ctl-C, CLEANDRV.BAT ends without performing the deletions. Pressing any other key allows the batch file to continue and, in line 45, DELELIST.BAT—the batch file created by CLEANDRV.BAT—finally is called to delete the unwanted files. Then line 46 neatens up the hard drive by deleting DELELIST.BAT and DELELIST.BAK (created when Edlin modifies DELELIST.BAT) before relinquishing control to CLEANALL.BAT.

Note: If you press Ctl-C to end the batch file before the deletions are done, the files DELELIST.BAT and DELELIST.BAK will remain on your hard drive. They will do no harm and will be overwritten the next time you run CLEANALL.BAT and CLEANDRV.BAT, but you may want to manually delete them for the sake of neatness.

When CLEANDRV.BAT concludes, control reverts to CLEANALL.BAT. Then, if CLEANALL.BAT contains another CLEANDRV command line with a different drive letter, CLEANDRV.BAT does it all over again, but this time acting on the different drive.

Uppercase Caveat

That's speed housekeeping under DOS 5.0 with an able assist from Edlin—a program you thought you'd never hear from again.

One final note: If you have a single hard drive, you can execute CLEANDRV.BAT directly by typing:

```
CLEANDRV C
```

and pressing Enter, instead of using CLEANALL.BAT. Be aware, though, that CLEANALL.BAT—in addition to allowing the cleanup of multiple drives in one operation—also passes uppercase drive letters to CLEANDRV.BAT, a requirement for CLEANDRV.BAT to operate correctly. The batch facility is rather finicky in the matter of case, so if you try to directly execute CLEANDRV.BAT with the command:

```
CLEANDRV c
```

you won't get any action. The output from DIR, including the drive designators, is all in uppercase. If you execute CLEANDRV.BAT with a lowercase drive letter as a parameter, line 9 in Listing 2 will set up a search-and-replace comparison that isn't going to find anything.

In short, I recommend that you always run CLEANALL to call CLEANDRV, even if you currently have only one hard drive. Then, if Santa is good to you next December, it will take only a few changes to include commands for your new drives in CLEANALL.BAT., and you'll be all set. Just be sure the drive letters you add are in uppercase. ■

Computer consultant Stephen Nebel resides in Fairbanks, Alaska.

Program Listing 2. CLEANDRV.BAT sweeps subdirectories clear of unwanted files. Lines 4, 5, 6, and 7 specify the types of files you want deleted from your hard drives. Depending on what you want to delete, you may need more or fewer lines in this section. (The line numbers are included as reference points; do not type them when you enter the listing into your text editor or word processor.)

```
 1 @ECHO OFF
 2 ECHO Working ...
 3 REM --- assemble list of files for deletion ---
 4 DIR %1:\*.BAK /S /B > DELELIST.BAT
 5 DIR %1:\*.OBJ /S /B >> DELELIST.BAT
 6 DIR %1:\*.RFT /S /B >> DELELIST.BAT
 7 DIR %1:\*.$?? /S /B >> DELELIST.BAT
 8 REM --- create segments for edlin script file ---
 9 ECHO 1,64000R%1:> EDLINCM1.ASC
10 ECHO E> EDLNEXIT.ASC
11 EDLIN EDLINCM2.ASC < EDLNEXIT.ASC > NUL
12 DEL EDLNEXIT.ASC
13 ECHO DEL %1:> EDLINCM3.ASC
14 ECHO E> EDLINCM4.ASC
15 REM --- create segments for debug script file ---
16 ECHO RCX> DEBUGCM1.ASC
17 ECHO A> DEBUGCM2.ASC
18 ECHO W> DEBUGCM3.ASC
19 ECHO Q> DEBUGCM4.ASC
20 REM --- assemble debug script file ---
21 DEL DEBUGCMD.ASC
22 COPY DEBUGCM1.ASC /B DEBUGCMD.ASC
23 COPY DEBUGCMD.ASC /B + DEBUGCM2.ASC
24 COPY DEBUGCMD.ASC /B + DEBUGCM3.ASC
25 COPY DEBUGCMD.ASC /B + DEBUGCM4.ASC
26 REM --- use debug to trim segment one to ten bytes ---
27 DEBUG EDLINCM1.ASC < DEBUGCMD.ASC
28 REM --- ASSEMBLE EDLIN SCRIPT FILE ---
29 DEL EDLINCMD.ASC
30 COPY EDLINCM1.ASC /B EDLINCMD.ASC
31 COPY EDLINCMD.ASC /B + EDLINCM2.ASC
32 COPY EDLINCMD.ASC /B + EDLINCM3.ASC
33 COPY EDLINCMD.ASC /B + EDLINCM4.ASC
34 REM --- call edlin in script mode to modify file ---
35 EDLIN DELELIST.BAT < EDLINCMD.ASC > NUL
36 DEL DEBUGCM*.ASC
37 DEL EDLINCM*.ASC
38 ECHO.
39 ECHO *** Here is the list of deletions to be performed:
40 TYPE DELELIST.BAT|MORE
41 ECHO.
42 ECHO To abort without carrying out
43 ECHO the listed deletions, press Ctl-C now.
44 ECHO Otherwise -
45 PAUSE
46 CALL DELELIST.BAT
47 DEL DELELIST.*
```

End

Q&A STARTING UP DOS

Q While deleting and undeleting files with Central Point Software's PC Tools, I somehow lost my hard drive. I cannot boot from drive C. I can, however, boot from my floppy drive, type:

```
C:
```

and then use the hard drive normally. Do I have to reformat my hard drive?

A A few different things might have happened to your hard drive, but most likely you simply deleted one of the three files that DOS needs to boot up and run. If you have not yet added any new files to drive C, and if you haven't run a file defragmentation program on the drive, the solution is simple: Copy the files from your bootup floppy and try to reboot. But if you have performed either of the above procedures, you may have to reformat your drive.

Here is the step-by-step process you should follow (assuming that your bootup floppy disk was made with the version of DOS you had on your hard drive). First, boot up from the floppy. Switch to drive C and then run a commercial file utility package such as PC Tools.

Next, look at all the files on your floppy disk. You should see a pair of files that have their hidden, system, and read-only attributes set. Their names should be something similar to IO.SYS and MSDOS.SYS (although both names will begin with IBM if you are using PC-DOS). Copy these two files to the root directory of your hard drive and, again, make sure that their attributes are set as hidden, system, and read-only. Also, copy the nonhidden file called COMMAND.COM to the same root directory—DOS needs all three files to boot up.

Try to boot from your hard drive. If it works, you're in business. If it doesn't, it's time to back up everything on the disk and then reinstall DOS on your hard drive. Depending on the version of DOS you are trying to install, you may have to use FORMAT before you can perform the installation.

One final note: The attributes on the DOS files are set to hidden, system, and read-only to protect them from accidental erasure. You would have had to override those settings to erase the files.

Q I have half-a-dozen subdirectories, separated by semicolons, in the PATH statement in my AUTOEXEC.BAT file, and all seem fine to me. But, I've noticed that many other peoples AUTOEXEC.BAT files are set up with a `PATH=%PATH%` statement. Am I doing this right? Are they?

A Both settings are right.

If you invoke SET at the DOS prompt, you will see a list of text on the screen. Each line starts with a word—*PATH*, for example—followed by an equal sign and then more text. This text is called the environment. On each line, the word to the left of the equal sign is called an environment variable, and the text to the right is an environment string, or the value of the variable. If you have a PATH statement in your AUTOEXEC.BAT file, one of the variables in the environment will be PATH.

When COMMAND.COM reads and executes each line in a batch file (including AUTOEXEC.BAT), it scans the line for words enclosed by percent signs. When it finds such a word, it looks for a matching variable in the environment. If it finds one, it substitutes the value of the variable (the variable string) for the variable name in the batch file.

Suppose, for example, you have a batch file containing the following two lines:

```
PATH C:\ABC
PATH %PATH%;C:\XYZ
```

The first line creates an environment variable called PATH and gives it a value of C:\ABC. When the second line is executed, COMMAND.COM looks up PATH in the environment and changes the line to read:

```
PATH C:\ABC;C:\XYZ
```

It then executes that line, establishing a new search path.

Some people use the `PATH %PATH%` syntax because it lets them more easily understand and modify their AUTOEXEC.BAT file; others like to put everything in one line. There is no advantage either way, and neither technique lets you extend the PATH beyond the maximum length of 127 characters (including the five characters in `PATH=`). ■

BY HARDIN BROTHERS

FILES AND DIRECTORIES Q&A

BY JAMES LANGDELL

Q I have just started downloading files on-line to collect software. I have a few files labeled with .ZIP extensions, but I can't run them even though I managed to download them successfully. How do I use these files?

A The file extension you mention indicates that the files are in a compressed format. On-line services commonly offer downloadable files in compressed form because smaller files transfer faster. Before you can make use of compressed files, however, you must expand them, using a utility such as the shareware program PKUNZIP. When they are decompressed, the files' correct file extensions will replace the .ZIP extension.

Most on-line services that offer compressed files also provide a program that will expand the files, along with a program you can use to compress files yourself. Documentation files should accompany the utility, but the program probably will start if you type the program name followed by the name of the compressed file.

A source for compression utilities is PKWARE Inc., 9025 N. Deerwood Dr., Brown Deer, WI 53223, 414-354-8699. The company operates a bulletin board service at 414-354-8670.

Q My job requires that I save a lot of small files. I have high-density floppy disk that contains over 220 files in a megabyte of space. By my estimate, nearly 400K of space remains, but in trying to copy more files onto the disk I get a "Cannot make directory entry" error message. What's wrong?

A DOS places a limit on the number of files you may store in the root directory of a floppy disk or a hard disk. For your high-density floppy disk, the limit is about 224 files. DOS imposes this limit for the root directory because it sets aside a specific area of each disk for the root directory. No provision for enlarging this area exists.

You can solve the problem by creating a subdirectory on your floppy disk and storing your numerous files in it. Information for a subdirectory, unlike information on the root directory, is held in a file that has no size limits. A subdirectory may hold as many files as your disk can accommodate.

Q My root directory contains three hidden files that take up over 50K of space. Are these so-called internal commands? Do they reside in memory when I turn on my computer? What is their purpose, and why are they hidden?

A Two of the hidden files are the system, or bootup, files that form the central part of DOS, which starts up automatically when you boot your computer. They load the resident portion of DOS into your computer's memory. The system files have hidden names that DOS 5.0's ATTRIB command can display. They are named either IBMBIO.COM and IBMDOS.COM (in PC-DOS) or IO.SYS and MSDOS.SYS (in MS-DOS).

DOS hides the files to reduce the chance of your entering commands that would accidentally corrupt them; this would prevent DOS from working. You shouldn't have to deal directly with those files because they are installed automatically when you format a disk to be bootable. These two hidden files can be reinstalled using the SYS command.

These two hidden files are not part of the DOS kernel, which means they are external files. The code that enables internal commands, such as DIR, COPY, and DEL, to work is contained in the unhidden COMMAND.COM file, a program sometimes called the command interpreter.

What about your third hidden file? That's the portion of your disk that contains the volume label that you specified either when you formatted the disk or, afterwards, using the LABEL command. You can see the name held in this hidden file by entering the command:

```
VOL
```

or looking for the volume name in a directory listing. ■

Q&A MEMORY MANAGEMENT

BY BRETT GLASS

Q I am still confused about expanded and extended memory. Which applications need which kind of memory? I've read that some spreadsheets "automatically" make use of expanded memory. I use a spreadsheet, and I have an 80286 system with 1MB of RAM. Do I have expanded memory?

A It is impossible to tell from the *amount* of memory in your machine whether you have expanded or extended memory. (Any machine with less than 640K of RAM, however, is unlikely to have either.) And the specific motherboard in your system determines whether the 384K of RAM that isn't used by DOS—that figure will be higher if you add memory to the motherboard—is available as extended or expanded memory. (Some motherboards waste the remainder, while others use it as a place to more quickly execute ROM BIOS routines in RAM.)

If you have a 386-based system, any 386 memory manager will let you make both extended and expanded memory available. In fact, some of the better memory managers convert one memory type to the other as needed. For people like you, who have a 286, getting extended memory isn't a problem. But you may need an expanded memory board if your motherboard doesn't have a Chips and Technologies NeAT or LeAP chip set.

As for applications, every one is different, and the only way to tell which kind of memory a program can use is by reading the product's manual. Even different programs within the same line vary: The latest versions of Lotus 1-2-3, for instance, work with extended memory, while some older ones can work only with expanded memory.

Q If the purpose of EMM386.EXE in MS-DOS 5.0 is emulating expanded memory, why don't computer makers convert all the extended memory in their systems into expanded memory?

A As the wheels of progress grind forward, more and more applications—especially ones that take advantage of advanced microprocessors such as the 386 and 486—run in Protected mode. Protected mode applications—including Windows 3.1—require extended memory. If all the RAM in your computer were expanded memory, you wouldn't be able to run these programs.

In the future, fewer new programs will use expanded memory. But because many older programs need it, an Expanded Memory Specification (EMS) emulator (also called a LIMulator) lets them run even on the newest, most powerful systems.

Q I have 2MB of RAM in my PC, but the MEM command reveals that I have 655,360 bytes of memory available to MS-DOS, 1,048,576 bytes of contiguous expanded memory, and zero bytes of available contiguous extended memory. This doesn't add up to 2MB, and I don't seem to have any extended memory. I am using DOS 4.01 and apparently the vendor of my system installed Microsoft's SMARTDRV.SYS disk caching program, though I'm not sure how it works. How do I get some of my extended memory back?

A I can think of two possible reasons why you don't see a total of 2MB of RAM in the MEM command's report. First, perhaps your system is using 384K of RAM to "shadow" your ROMs, copying the contents of the ROMs, which are slow, into RAM, which is faster. Second, you may have a motherboard that, because its designers were lazy, wastes 384K of the first megabyte of RAM. These motherboards aren't as common as they once were, but many older 286 systems have them.

There are several reason why you might have no bytes of available contiguous memory. The most likely reason is that an extended memory standard (XMS) driver, such as Microsoft's HIMEM.SYS, is installed on your system. An XMS driver converts "raw" extended memory into XMS memory. This doesn't mean it's no longer available for use; in fact, many programs that use extended memory, such as Windows, can't use memory unless it's XMS memory.

Another possible reason why no extended memory remains is that it's monopolized by drivers or memory-resident programs, such as SMARTDRV.SYS. You can test for this by removing the line that invokes this program from your CONFIG.SYS file (or placing a REM command at the beginning of that line). ■

BATCH FILE PROGRAMMING Q&A

Q I am fascinated by batch file programming. Are there any commercial products that expand the list of batch file commands? I'm looking for a way to add a timed element to my batch files—some way to watch the clock—but I can't find one.

A The batch languages provided with MS-DOS 5.0 and DRDOS 6.0 don't provide the interactive command you need to check the computer's time-of-day clock. I suggest you try JP Software's 4DOS 4.0 or Symantec's NDOS (a program included in The Norton Utilities 6.0).

These products offer slightly different features, but both are derived from JP Software's replacement for the MS-DOS command interpreter. At start-up, 4DOS replaces the DOS command interpreter (COMMAND.COM) with its own, more sophisticated version. The added power and flexibility these two software packages provide are too numerous to list here. For the ardent batch programmer, however, owning a copy of either is a must.

The $69 commercial version of 4DOS 4.0 is available directly from JP Software (P.O. Box 1470, East Arlington, MA 02174, 617-646-3975 or, for orders only, 800-368-8777). If you would like to try 4DOS before buying it, a shareware version is available on many computer bulletin boards. You can find it on the CompuServe Information Service in the PC Vendor B Forum. To get there, type GO JPSOFT at any ! prompt. You can find The Norton Utilities ($179) package at most software stores and through mail-order vendors.

Q Can I write batch files that work from within an application? Could I use them like macros?

A The answer is no, unless your application includes a feature that lets you execute DOS batch files directly from within its own program code.

After an application program loads, it takes control of your computer, which makes it impossible to interact directly with DOS without the aid of sophisticated programming techniques or special memory-resident programs.

Many applications, such as word processors, communications programs, and database managers provide macro languages that let you execute internal program functions using a script-like language. Although these script languages may look like DOS batch files, they're generally used only to execute functions built into the program.

Other programs use the DOS SHELL command to let you exit to the DOS prompt to execute DOS batch files and other DOS commands. But your applications effectively "go to sleep" while you work at the DOS prompt. So, you cannot combine actions that occur at the DOS prompt with those from within your application program.

Q I know a little about programming, but the batch language in DOS throws me. It has an IF statement but no THEN to go with it. It also seems to lack a NOT statement. How then is decision-making accomplished in a batch file? Can you give me a couple of examples?

A Although no THEN key word exists in the DOS batch language, the IF command implies it. For example, the following batch file command:

```
IF EXIST MYFILE.DAT ECHO
  Your file is here.
```

is functionally the same as this QBasic command:

```
IF EXIST MYFILE.DAT THEN ECHO
  Your file is here.
```

In addition, although the DOS batch language doesn't include a specific NOT command, the IF command lets you include the optional NOT operator. In the first example above, the IF command uses this syntax:

```
IF EXIST [filename] [command]
```

By default, the IF batch command will execute the command parameter only when MYFILE.DAT exists in the current DOS directory. If you include the optional NOT keyword, however, the condition is logically reversed:

```
IF NOT EXIST MYFILE.DAT ECHO
  Your file isn't here.
```

The syntax is as follows:

```
IF NOT EXIST [filename] [command]
```

In this case, the command parameter is executed only when the file doesn't exist in the current directory. ■

BY JOHN WOLFSKILL

Q&A HARDWARE

Q I have a computer monitor that supposedly is capable of displaying my applications at 1024- by 768-pixel resolution, but my software looks no different to me.

A Although your monitor may be capable of displaying your application program screens with 768 lines of 1024 graphic dots each, it won't do so automatically. Your application program or operating environment (such as Microsoft Windows) must be designed to support high-resolution text and graphics. Typically, you accomplish this by loading a special software device driver prior to running your application or by selecting a special graphics operating mode from within the application's configuration menu.

The increased resolution your monitor can deliver also requires a special Super VGA (SVGA) graphics adapter before it can display your programs at resolutions greater than 640 by 480 pixels, which is the maximum a standard IBM-compatible VGA graphics card can provide.

Programs designed to operate in any one of the standard IBM text modes generally won't take full advantage of a high-resolution monitor such as yours.

Q As a prospective PC buyer, what do FCC ratings mean to me?

A The Federal Communications Commission (FCC) requires manufacturers of electronic equipment to submit representative samples for certification testing before their computers may be offered for sale. The purpose of this testing is to ensure that the equipment doesn't leak excessive amounts of radio-frequency energy into the atmosphere, which can create a potentially troublesome problem, called radio-frequency interference (RFI).

Almost all electronic equipment emits some RF waves, electronic gremlins that can wreak havoc with your neighbor's television reception, zap sensitive airport runway equipment, and cause commercial radio transmissions to go out of kilter. The FCC, however, requires that computer manufacturers design their machines to limit those emissions.

Although a PC manufacturer is responsible (and liable) for failure to comply with FCC regulations, the ultimate responsibility for infraction of FCC rules rests with the purchaser. If your PC spews too many harmful RF waves into the air, and someone complains about it, the FCC can obtain a court order to seize your equipment.

The FCC prescribes several classification levels for computing equipment. A Class A rating (the minimum) entitles the manufacturer to sell equipment for use in business settings only. If the equipment passes a more stringent test, it receives a Class B rating, which clears it for use in residential areas.

To make sure the PC you're thinking of buying is appropriate for the setting in which you plan to use it, check the rear of the computer for the small FCC sticker. Printed on the sticker you'll find a number that must be at least seven but not more than 17 digits long. FCC identification numbers that end with the letter A indicate that the equipment's FCC status has been amended since its original test.

If you don't see the sticker, ask the salesperson to show you the official FCC statement page, which *must* appear in the documentation supplied with the equipment. If you suspect the registration number is bogus, you can call the FCC's Public Access Link (PAL) BBS, via modem, at 301-725-1072 (300/1200 bps). PAL lets you check a registration number against those on file in the FCC's official database.

Q My computer is equipped with an IDE drive. Can I add a SCSI interface card with no ill effects? I'm thinking about installing a CD-ROM drive in my system in the future.

A You can use both devices, providing they are configured to use different CPU ports (areas where hardware and software place data for DOS to load) and separate hardware interrupt request lines (IRQs) to signal the central processor that they need to process information. All SCSI (Small Computer Systems Interface) host adapter cards come with small jumper plugs that let you change the setting of interrupt and CPU port assignments so that the new device won't conflict with devices already in your system. In most cases, you won't need to change the settings, however, because the factory settings take into consideration the standard devices in most PC systems. ■

BY JOHN WOLFSKILL

Listing continued from p. 62

```
      GOSUB DISPLAYMESSAGE 'and display it
      FF = 0               'set function flag to "action"
      MAXPOS = 26          'assign maximum cursor column position
      MINPOS = 5           'assign minimum cursor column position
      GOSUB EDITOR         'call the string editor
      EXIT DO              'exit this loop

     CASE "C"              'if the C key was pressed
      MESSAGE = 5          'select a message
      GOSUB DISPLAYMESSAGE 'and display it
      FF = 1               'set the function flag to "control"
      MAXPOS = 79          'assign maximum cursor column position
      MINPOS = 30          'assign minimum cursor column position
      GOSUB EDITOR         'call the string editor
      EXIT DO              'exit this loop

     CASE "E"              'if the E key was pressed
      MESSAGE = 1          'select a message
      GOSUB DISPLAYMESSAGE 'and display it

     CASE ELSE
      END SELECT           'end of key case structure
 LOOP                      'bottom of select function loop

'(M) - Action and control-string assignment routine -
 IF CANCELKEY = 0 THEN    'if Esc pressed, bypass actions below
  IF FF = 0 THEN          'if function flag set for "action" strings
    ACTION$(CLINE) = G$   'assign G$ to action string variable
  ELSE                    'else function flag set for "control" strings
    CONTROL$(CLINE) = G$  'assign G$ to control string variable
  END IF
 END IF

 GOSUB GATHER             'read and display the input
 MESSAGE = 1                'select a message
 GOSUB DISPLAYMESSAGE   'then display it
 RETURN                     'exit this routine (return to the main prompt)

'(N) - Action string editing routine -
EDITOR:

 MAXLINE = 17                  'define the bottom line of input box
 MINLINE = 6                   'define the top line of input box
```

Listing continued ➛

➛ Listing continued

```
 CPOS = MINPOS                'set the column counter to leftmost position
 CLINE = MINLINE              'set the line counter to the top of the box

 LOCATE MINLINE, MINPOS       'home the cursor, top line of box
 GOSUB GATHER                 'read the screen, print the message
 COLOR 0, 7                   'set the highlight colors (black on white)
 LOCATE MINLINE, MINPOS       'reset the cursor to leftmost position
 GOSUB GATHER                 'read the screen, print the message

'(O) - String editing routine -
 DO                           'top of the editing loop:
 X$ = INKEY$                  'check for a keystroke
    SELECT CASE (X$)          'define a key case structure:

     CASE ""                  'if no key is being pressed
      GOSUB FLASH             'flash the cursor

     CASE CHR$(13)            'if the Enter key was pressed
      GOSUB GATHER            'read and display the text input
      RETURN                  'exit this routine

     CASE CHR$(27)            'if the Esc key was pressed
      CANCELKEY = 1           'set the cancel flag
      RETURN                  'exit this routine

     CASE UPKEY$              'if the up-arrow key was pressed
      IF CLINE > MINLINE THEN 'then if line is within range
       COLOR 7, 0             'set colors (white on black)
       GOSUB GATHER           'read the screen, print the message
       CLINE = CLINE - 1      'move the cursor bar up
       COLOR 0, 7             'set highlight colors (black on white)
       GOSUB GATHER           'read the screen, print the message
      END IF

    CASE DOWNKEY$             'if the down-arrow key was pressed
    IF CLINE < MAXLINE THEN   'then if the line is within range
     COLOR 7, 0               'set colors (white on black)
     GOSUB GATHER             'read the screen, print the message
     CLINE = CLINE + 1        'move the cursor bar down
     COLOR 0, 7               'set highlight colors (black on white)
     GOSUB GATHER             'read the screen, print the message
    END IF
```

Listing continued on p. 76

Listing continued from p. 75

```
    CASE LEFTKEY$               'if the left-arrow key was pressed
     IF CPOS > MINPOS THEN      'then if cursor value is within range
      CPOS = CPOS - 1           'move the cursor left
     END IF

     CASE RIGHTKEY$             'if the right-arrow key was pressed
      IF CPOS < MAXPOS THEN     'then if cursor value is within range
       CPOS = CPOS + 1          'move the cursor right
      END IF

     CASE CHR$(8)               'if the backspace key was pressed
      IF CPOS > MINPOS THEN     'then if cursor value is within range
       LOCATE CLINE, CPOS: PRINT " "; 'erase the character under cursor
       CPOS = CPOS - 1          'move the cursor left
      END IF
     CASE HOME$                 'if the Home key was pressed
      LOCATE CLINE, MINPOS      'set the cursor to leftmost position
      CPOS = MINPOS             'set cursor counter to minimum value

     CASE END$                  'if the End key was pressed
      LOCATE CLINE, MAXPOS      'set cursor to rightmost position
      CPOS = MAXPOS             'set cursor counter to maximum value

     CASE CHR$(32) TO CHR$(127)'if any other typing key was pressed
      LOCATE CLINE, CPOS        'locate the cursor
      PRINT X$;                 'print the character
      IF CPOS < MAXPOS THEN     'if cursor is within range
        CPOS = CPOS + 1         'move the cursor right
      END IF

    CASE ELSE
                END SELECT               'end of the key case structure

   LOOP                         'bottom of editing loop

 '(P)- Send control strings to printer -
SEND:
  MESSAGE = 3                   'select a message
  GOSUB DISPLAYMESSAGE          'display it
  CLINE = LASTITEM              'set to previous item choice
  IF CLINE < 6 OR CLINE > 17 THEN CLINE = 6 'check range
  COLOR 0, 7                    'set highlight colors (black on white)
```

Listing continued

```
  GOSUB GATHER                  'and display the selected letter

 DO                             'top of end key loop:
  X$ = INKEY$                   'check for a keystroke
    X$ = UCASE$(X$)             'force the ASCII value to uppercase
     SELECT CASE (X$)           'define a key case structure:

        CASE CHR$(65) TO CHR$(76)'if keystroke within range A-L then
         COLOR 7, 0             'set colors (white on black)
         GOSUB GATHER           'and display the letter
         CLINE = ASC(X$) - 59   'calculate the display line
         LASTITEM = CLINE       'save choice for next time
         COLOR 0, 7             'set highlight colors (black on white)
         GOSUB GATHER           'and display the selected letter

        CASE CHR$(13)           'if the Enter key is pressed then
           GOSUB SENDCODES      'send the string to the printer
           RETURN               'exit this routine

        CASE CHR$(27)           'if the Esc key is pressed then
           RETURN               'exit this routine

       CASE ELSE
        END SELECT              'end of the key case structure

   LOOP                         'bottom of send key loop

 '(Q) - Send control string to printer -
SENDCODES:

  J = 0                         'zero the string character counter
  B$ = ""                       'set up a temporary string variable
  C$ = RTRIM$(LTRIM$(CONTROL$(CLINE))) + ","   'select a control string

DO
  J = J + 1                     'increment the string character counter

  IF J > LEN(C$) THEN EXIT DO   'if we've reached the end, exit this routine
   A$ = MID$(C$, J, 1)          'read the next character
                                '
   IF A$ = "#" THEN             'if it's a "#" then print the test string
      LPRINT MID$(C$, 2, LEN(C$) - 2); CHR$(13); CHR$(10)
```

Listing continued on p. 77

Listing continued from p. 76

```
      EXIT DO                       'exit this loop
    END IF

   IF A$ = "," OR J = LEN(C$) THEN    'if it's a comma or end of string
      LPRINT CHR$(VAL(B$)); : B$ = "" 'print the string
    ELSE                            'otherwise
      B$ = B$ + A$                  'add its value to the temp string
   END IF

  LOOP                              'and loop for another character

  RETURN                            'exit this routine (to the main prompt)

'(R) - Gather input from screen  -
GATHER:

  G$ = ""                           'initialize a temporary string variable

  FOR NPOS = MINPOS TO MAXPOS       'read characters from the screen
   G$ = G$ + CHR$(SCREEN(CLINE, NPOS))  'use the temp variable to hold them
  NEXT

  LOCATE CLINE, MINPOS              'locate the cursor
  PRINT G$;                         'print the string
  RETURN                            'exit this routine

'(S) - Display the data items -
DISPLAYDATA:

  FOR J = 6 TO 17                   'top of counter loop:
   LOCATE J, 5                      'position the cursor
   PRINT ACTION$(J);                'display the action strings
   LOCATE J, 30                     'position the cursor
   PRINT CONTROL$(J);               'display the control strings
  NEXT                              'bottom of counter loop:
 RETURN

 '(T) - Flash the cursor -
FLASH:

  FL$ = CHR$(SCREEN(CLINE, CPOS))'read the character under the cursor
  COLOR 7, 0                        'set colors (white on black)
  LOCATE CLINE, CPOS                'set the cursor position
```

Listing continued →

→ Listing continued

```
   PRINT CHR$(219);                 'print a big white block
   GOSUB PAUSE                      'wait a few clock ticks
   LOCATE CLINE, CPOS               'locate cursor
   PRINT FL$;                       'replace the original character
   GOSUB PAUSE                      'wait a few clock ticks
   RETURN                           'exit this routine

 '(U) - Pause timer -
 PAUSE:

   T! = TIMER                       'get the current time
   WHILE T! = TIMER: WEND           'execute a do-nothing loop
   RETURN                           'exit this routine

 '(V) - Print the user prompt box -
 DISPLAYMESSAGE:

   LOCATE 20, 20                    'locate cursor
   PRINT CHR$(201) + STRING$(36, 205) + CHR$(187) 'display a string
   LOCATE 21, 20                                  'locate cursor
   PRINT CHR$(186) + STRING$(36, 32) + CHR$(186)  'display a string
   LOCATE 22, 20                                  'locate cursor
   PRINT CHR$(200) + STRING$(36, 205) + CHR$(188) 'display a string
   LOCATE 21, 21                    'locate cursor
   PRINT MESSAGE$(MESSAGE);         'print the message
   RETURN                           'exit this routine

 '(W)  - Save data file to disk -
 SAVEFILE:

   OPEN "PP.DAT" FOR OUTPUT AS #1 'open the data file for output
    FOR J = 6 TO 17               'top of counter loop:
     WRITE #1, CONTROL$(J)        'write the control strings to disk
     WRITE #1, ACTION$(J)         'write the action strings to disk
    NEXT                          'bottom of counter loop
   CLOSE 1                        'close the file handle
   RETURN                         'exit this routine

  '(X) - Load data file from disk -

 LOADFILE:

 9999  OPEN "PP.DAT" FOR INPUT AS #1  'open the data file for input
```

Listing continued on p. 78

Listing continued from p. 77

```
        FOR J = 6 TO 17              'top of counter loop:
        INPUT #1, CONTROL$(J)        'read control strings from disk
        INPUT #1, ACTION$(J)         'read action strings from disk
        NEXT                         'bottom of counter loop
      CLOSE 1                        'close the file handle
      RETURN                         'exit this routine

 '(Y) – Display the program screen –
INITSCREEN:

 COLOR 7, 4                      'set colors (white on red)
 LOCATE 1, 1: PRINT STRING$(80, 32);              'print a blank line
 LOCATE 1, 26: PRINT "DOS RESOURCE PRINTER PRIMER"; 'display a string

 '–Display the action string box –

 COLOR 0, 7                                       'set colors (black on white)
 LOCATE 3, 1                                      'locate cursor
 PRINT CHR$(201) + STRING$(25, 205) + CHR$(187) 'display a string
 LOCATE 4, 1                                      'locate cursor
 PRINT CHR$(186) + STRING$(9, 32) + "ACTION" + STRING$(10, 32) + CHR$(186)
          LOCATE 5, 1                                       'locate cursor
 PRINT CHR$(199) + STRING$(2, 196) + CHR$(194) + STRING$(22, 196) + CHR$(182)

 '– Display the item identifiers –

 SPOS = 1                          'begin with first character

 FOR LNE = 6 TO 17                 'top of counter loop:
  LOCATE LNE, 1                    'locate cursor
  PRINT CHR$(186); MID$(NUM$, SPOS, 2); 'print portion of string
  PRINT CHR$(179) + STRING$(22, 32) + CHR$(186); 'print the rest
  SPOS = SPOS + 2                                'increment counter
 NEXT                                            'bottom of counter loop

 LOCATE LNE, 1                     'locate cursor
 PRINT CHR$(200) + STRING$(2, 205) + CHR$(207) + STRING$(22, 205) + CHR$(188);
 COLOR 7, 1                        'set colors (white on blue)

 ' – Display the control string box –

 COLOR 0, 7                        'set highlight colors (black on white)
```

Listing continued →

→ Listing continued

```
 LOCATE 3, 29                                        'locate cursor
 PRINT CHR$(201) + STRING$(50, 205) + CHR$(187)  'display a string
 LOCATE 4, 29                          '             'locate cursor
' Note: type the following two lines as one program line.
PRINT CHR$(186) + STRING$(16, 32) + "CONTROL STRING" + STRING$(20, 32) +
  CHR$(186)
 LOCATE 5, 29                          '             'locate cursor
 PRINT CHR$(199) + STRING$(50, 196) + CHR$(182)  'display a string

 ' – Display the body –

 FOR LNE = 6 TO 17                    'top of counter loop
   LOCATE LNE, 29                     'locate cursor
   PRINT CHR$(186) + STRING$(50, 32) + CHR$(186); 'display a string
  NEXT                                             'bottom of counter loop

 '– Display the bottom border –

  LOCATE 18, 29                       'locate cursor
  PRINT CHR$(200) + STRING$(50, 205) + CHR$(188);  'display a string
  RETURN                                            'exit this routine

 '(Z) – Error Handler –
TESTERROR:

  IF ERL = 9999 AND ERR = 53 THEN
   GOSUB SAVEFILE: RESUME LOADFILE  'if data file doesn't exist, create it
  END IF

  COLOR 7, 0: CLS : BEEP              'clear the screen. Red alert
  PRINT "Error "; ERR; " in line "; ERL; 'print the error line and number
  CLOSE                               'just in case, close all files
  END                                 'Oops! End the program

 '—- End of program code ———
End
```

End

Now available to readers of *DOS Resource* from

TECHMEDIA DISCOUNT

A quality selection of entertainment software (see next page for our other products) at incredibly low prices! We buy aggressively and pass the savings on to our customers. Compare our prices on these products to any other retailer!

Live Studios

Future Classics Collection
5 great games in one package! Lost'n Maze, Diet Riot, DiskMan, TankBattle, and BlockAlanche. #TDG1.
List Price $39.95
Sale Price $19.95

Thunderstrike
Futuristic, 3-D flight simulation experience. Compete in the deadly military olympics. #TDG2
List Price $39.95
Sale Price $21.95

Psygnosis

Atomino
Brain teaser with over 60,000 levels. Form molecules from atoms as they spontaneously generate in your test tube. #TDP1. List Price $49.95
Sale Price $29.95

Obitus
Massive, role-playing arcade/adventure with silky-smooth 3-D scrolling, complex puzzles, character interaction, arcade action and multiple playing styles. #TDP2
List Price $49.95
Sale Price $29.95

Never Mind
Mind bending puzzle game with 300 levels. #TDP3 List Price $39.95
Sale Price $14.95

Hollyware

Faery Tale Adventure
Graphic fantasy role playing. Guide 3 brothers on their quest to save the land of Holm from the evil forces of the night. #TDN1 List Price $49.95
Sale Price $19.95

Faery Tale Clue Book
Large wall map and detailed maps for exploring the realm of Holm. #TDN2 List Price $9.98
Sale Price $5.95

Romantic Encounters
Adult text experience. Interact, as a woman or a man, with enticing scenarios waiting for you at the Dome. #TDN3 List Price $24.98
Sale Price $11.95

MicroProse Software

Famous for simulation software!

Gunship
Be put in the cockpit of the awesome AH-64A Apache attack helicopter. #TDM1 List Price $19.95
Sale Price $11.95

Silent Service
Popularly acclaimed as the definitive World War II submarine simulation. #TDM2 List Price $19.95
Sale Price $11.95

F-15 Strike Eagle
Strap yourself into the cockpit of one of the most powerful and technologically advanced fighter planes and engage in combat. #TDM3
List Price $19.95
Sale Price $11.95

Solo Flight
Learn to fly. Full instrumentation and maps let you navigate electronically among over 20 U.S. cities. #TDM4
List Price $19.95
Sale Price $11.95

Star Glider II
Interplanetary combat adventure. Invade the Empire's most heavily-guarded solar system and knock out the space station. #TDM5
List Price $19.95
Sale Price $11.95

Airborne Ranger
Be one of America's elite soldiers. Embark on solo missions behind enemy lines and rescue hostages from terrorists. #TDM6
List Price $44.95
Sale Price $14.95

3-D Pool
Test your skill with 15 different trick shots, or create your own. Then enter tournament play against formidable opponents.#TDM7 List Price $19.95
Sale Price $11.95

Dr. Doom's Revenge
Spiderman and Captain America team up to combat Dr. Doom and his villains. Navigate through Dr.Doom's castle and disarm the missile. #TDM8 List price $19.95
Sale Price $11.95

Midwinter
Compelling strategy game. Control up to 32 characters and defend 160,000 square miles.Ski, hang-glide, snow-buggy or travel by cable car. #TDM9 List Price $19.95
Sale Price $11.95

Space 1889
Mystery/role-playing adventure combining history and classic science fiction. Overcome corrupt conspirators, a Martian religious cult and an advanced alien society. #TDM10
List price $19.95
Sale price $11.95

Savage
Arcade/fantasy action game pitting the player against a variety of magical, evil creatures in a bizarre, castle fortress. #TDM11
List Price $19.95
Sale Price $11.95

Troika
Trilogy of addictive arcade/strategy masterpieces #TDM12 List Price $39.95 Sale Price **$18.95**

Accolade

Gunboat
Operate a U.S. Navy River Patrol Boat in over 20 missions #TDA1
List Price $49.95
Sale Price $24.95

Heatwave:Offshore Superboat Racing
Puts you at the helm of 4 superboats as you compete on a circuit of realistic, 3-D courses. #TDA2 List Price $49.95
Sale Price $22.95

Strike Aces
Pilot one of 6 high-speed strike aircraft against 7 international interceptors, embark on 16 different bombing missions. #TDA3
List Price $49.95
Sale Price $18.95

Digitek

New Releases!

Big Business
Humorous simulation of a large manufacturing corporation. #TDD1
List Price $49.95
Sale Price $27.95

Dino Wars
Interactive strategy arcade, features superb graphics and amazing digitized sounds #TDD2
List Price $39.95
Sale Price $24.95

Hole-in-One Miniature Golf
Sports simulation with 90 holes to play. #TDD3 List Price $34.95
Sale Price $24.95

Rings Of Medusa
Explore 33 cities, 13 castles and 3 islands as part of the quest to get your father's kingdom back from Medusa. #TDD4
List Price $49.95
Sale Price $27.95

Closeouts!

Limited quantities available.

Interplay

James Bond: Stealth Affair
#TDC1 List price $49.95.
Closeout Price $22.95

Cinemaware

Defender of the Crown
#TDC2 List Price $39.95.
Closeout Price $11.95

SDI
#TDC3 List Price $39.95.
Closeout Price $11.95

TV Sports Football
#TDC4 List Price $49.95.
Closeout Price $11.95

Dragon Lord
#TDC5 List Price $39.95.
Closeout Price $11.95

MicroDeal

Air Ball
#TDC6 List Price $39.95.
Closeout Price $12.95

To Order Call

1-800-824-5499

Call 410-546-0180 for Customer Service and orders outside the U.S. Use Code DR692 when ordering.

Most Major Credit Cards accepted. CA, MD and Canadian residents will need to add appropriate taxes. See following page for mailing order form.

When ordering please state whether you have a 3 1/2" or 5 1/4" disk drive. All games require a minimum of 512k to play.

TECHMEDIA DISCOUNT, PO Box 2151, Salisbury, MD, 21802

DR692

TECHMEDIA DISCOUNT

Our savings continue!

Abacus

Tips & Tricks for your PC Printer Learn to use your non-laser printer to its fullest potential with this handy guidebook. Includes companion disk packed with printer utilities. #TDB1 List price $34.95 **Only $24.95**

Laser Printer PowerTools A book full of information that lets you harness the power of your laser printer. Includes companion disk of printer drivers and utilities.#TDB2 List price $34.95 **Only $24.95**

Personal Portfolio Manager A comprehensive portfolio management and analysis software package for your PC. Provides automatic Dow Jones News/Retrieval updates on modem-equipped systems.#TDB3 List price $150.00 **Special price $69.95**

Tempest

Program manager and **DOS shell** for all DOS computers. Provides mouse support for all DOS commands.#TDB4 Just released! **Only $29.95**

Saddleback Graphics

My Paint The Paint Program Just For Kids! A Full-featured painting program with a computer coloring book included. MyPaint is easy to use, educator approved, classroom tested and packed with features. The built-in coloring book has 28 fascinating features, each with its own narration. #TDG3 List price $49.95 **Sale Price $34.95**

Additional coloring disks for My Paint

Alphabet Fun! Easy-to-color drawings of each letter of the alphabet, with accompanying digitized sounds. #TDG4 List price $19.95 **Sale Price $14.95**

Majelix Characters The Majelix character names provide children with a powerful reference to the English phonics code which helps them with their reading skills. #TDG4 List price $19.95

Special offer! #TDG6 Get both coloring disks for **only $24.95**

Sale Price $14.95

Productivity Disk from *TechMedia* Super Value! Make every computing operation go more smoothly and make yourself more productive when you employ CRUISER and the 14 other utility programs on this special disk. CRUISER allows you to see inside any file on your disk and page through them as if the disk were one large book. CRUISER has a text editor, pop-up calculator, and much more. #TDT1 **Special Price $22.95**

DeluxePaint Animation from *Electronic Arts* Two programs in one: 256-color paint and image processing tools plus powerful animation features. Paint features include stencil mode, translucency effects, fully customizable brushes, instant 3-D perspective, and much more. Animation features include metamorphosis effects, move requesters to define any movement, creation of animated brushes, color cycling, scrolling backgrounds, slide show utility, and more. System requirements are IBM P/S2 or compatible 286 or greater, 640K, MCGA or VGA, Microsoft compatible mouse, and hard drive.#TDE1 List price $134.00 **Sale Price $94.95**

DeluxePaint Animation Video from*Saddleback Graphics* This one-hour instructional video of tutorials, tips, techniques and interviews shows both the new user and the seasoned pro how to best utilize DeluxePaint Animation's tools to achieve spectacular effects with professional results. #TDG7

Special offer! #TDG8 Get both DeluxePaint Animation and the Video for **only $104.95**

List price $29.95 **Sale Price $24.95**

Top quality Modems from *Supra Corp.*

SupraModem 2400 Stand-alone 2400 baud #TDS1 **Sale Price $94.95**

SupraModem 2400 Plus 2400 baud with error-checking #TDS2 **Special Price $139.95**

Supra FAX Modem 2400/9600 Plus Includes fax capabilities #TDS3 **Sale Price $164.95**

Supra FAX Modem V32 bis Fax and data compression, up to 14,400 baud #TDS4 **Special Price $379.95**

SupraModem 2400i Internal 2400 baud modem #TDS5 **Special Price $74.95**

SupraModem 2400i Plus 2400 baud internal with error checking #TDS6 **Special Price $104.95**

Added Value! All Modem packages come with telecommunications soft ware and starter kit for America Online network. All external modems come with modem cables– an **$11.95 value!**

DB-25 Modem Cable For all external PC modems #TDH1 **Sale Price $11.95**

Parallel Printer Cable

Free Gift! Place an order for $75 or more and receive a free game of your choice from the preceding page. Place an order for $125 or more and receive two free games of your choice!

To Order Call

1-800-824-5499

For orders outside the U.S. call 410-546-0180. Please use Code DR692 when ordering.

Hours open: 8 A.M. to 6 P.M. Eastern Time For Technical Assistance call 603-924-0108

MAILING ORDER FORM

☐ **Check here for free game offer.**
List FREE game(s) below.
______________#______
______________#______

☐ Check Enclosed ☐ Charge My ☐ Mastercard ☐ VISA ☐ AmExpress ☐ Discover

Account # ______________
Expiration ______________
Signature ______________

Shipping Address
Name ______________
______________Address
______________City,State, Zip

Quantity	Item #	Item Name	Unit Price	Total Price
#			$	$
#			$	$
#			$	$
#			$	$
#			$	$
#			$	$
#			$	$
100% Money Back Guarantee		SUBTOTAL		$

Product Total $______
Add $3.95 Postage & Handling $______
(Canada & Mexico add $6.00 $______
Foreign add $11.50)
CA,MD and Can.(7% GST)
residents, add tax $______
Total Enclosed $______

Mail To: **TechMedia Discount**, PO Box 2151, Salisbury, MD, 21802. Please allow 3-4 weeks for delivery. Some product quantities are limited. All orders shipped via surface mail. Call for details on special shipping requirements. DR692

SUBSCRIBE TO DOS RESOURCE GUIDE 800-238-6580

AND SAVE 20% OFF

THE COVER PRICE

BUCK THE TREND! STAY WITH DOS

Subscribe NOW and get 4 issues of DOS RESOURCE GUIDE for $15.80. Save a buck an issue off the newsstand price!

If you're having trouble understanding DOS, stop worrying. It's not you!

Understanding DOS is the key to effective PC-computing. But if you've tried to learn DOS on your own, wrestled with its manual, purchased a guidebook, or taken a course, you're probably still frustrated and confused.

Relax, you're not alone!

One estimate puts the number of DOS-run computers in the marketplace at 70 million. But how many DOS experts are there?

Darn few, and here's why.

Since late 1980, Microsoft Corp.'s DOS, the essential software ingredient of the IBM PC-compatible , has changed many times. Along the way, the operating system that began as 22 cryptic commands has grown to about 100 equally cryptic commands, drivers, and programs.

DOS RESOURCE GUIDE explores DOS's hidden secrets, untangles the twisted technical prose of manuals, and tells you in clear language how to use DOS.

In each issue of DOS RESOURCE GUIDE, you'll find articles about using DOS commands, about managing your computer's memory, or maintaining order on your hard disk.

Beginners will find step-by-step help at copying files, creating and removing directories, using DOS's wildcards to cut down on keystrokes, and practical advice on using DOS's many "switches" after each command.

More advanced users will learn about batch files, which can provide capabilities that DOS forgot, or automate tedious activities, such as moving or backing up large groups of files. And you'll learn to use DOS 5's new QBasic programming language, and create macros at DOS's command line.

Each issue of DOS RESOURCE GUIDE is filled with tips on using DOS, undocumented shortcuts, and batch files you can type in and use.

So, tap all of your computing potential by learning DOS, personal computing's most important—if not most popular— operating system. Order today! Use the coupon below, or call:

800-238-6580

YES! I'M BUCKING THE TREND! YES! I'M STAYING WITH DOS!

☐**YES!** Send me 4 Issues of DOS Resource Guide for $15.80.

NAME ____________________
ADDRESS ____________________
CITY ____________________ STATE ________ ZIP ________

(Check, Money Order, or Credit Card ONLY)

☐ AMEX ☐ VISA ☐ MASTERCARD ☐ DISCOVER

CREDIT CARD # ____________________
EXPIRATION DATE: ____________________
SIGNATURE: ____________________

MAIL TO: DOS Resource Guide, 80 Elm Street, Peterborough, NH 03458

TAKE THE DREAD OUT OF DOS!

If you're trying to succeed with DOS, but feel like a General Failure...

DOS RESOURCE GUIDE IS WRITTEN FOR YOU

DOS RESOURCE GUIDE, the quarterly periodical packed with tips, tutorials, and information on all aspects of PC computing, is your antidote for computing confusion.

DOS RESOURCE GUIDE #1

Make DOS Work for You

Covers six areas of computing that are vital to DOS users: start-up files (AUTOEXEC.BAT and CONFIG.SYS), batch file creation, hard drives, safe backups, organizing file management, and hardware maintenance. Includes: List'em, a batch file that helps you keep appointments; 10 tips to keep your hard drive healthy; speed-up tips for a logy hard drive; a batch file that automates backups; DOS shortcuts, hardware dos and don'ts. Plus 42 provocative pointers on DOS.

DOS RESOURCE GUIDE #2

Guide to Windows Software

SOLD OUT

A compendium of software briefs that tells you what's up in the world of Windows. (Even a DOS user has to keep up with the competition.) Guide #2 briefly reviews over 140 programs for Microsoft Windows, including word processors, communications programs, financial and mathematical programs, management programs, developer's tools, and graphics software. Complete list of manufacturers.

DOS RESOURCE GUIDE #3

The Essential Guide to DOS

Who are we to argue with success! Our second edition of this classic is updated to include information on DOS 5. Guide #3 includes an annotated list of DOS files and when to use them; articles that will change your view of CHKDSK, COPY, and XCOPY; a cluster of batch file assists; and some clear-headed advice on organizing files on your hard drive. Plus 14 titillating tips on DOS.

DOS RESOURCE GUIDE #4

The Complete DOS Q&A

Breaks DOS into functional sections: Fundamentals, Set-up, Improving Productivity, Memory Management, Hardware Upgrades, Batch Files, and Using Windows. Each section is followed by a question and answer session on the topic. Plus, seven short programs to protect your data; do-it-yourself memory, advice on CONFIG.SYS and AUTOEXEC.BAT, and beginner's Basic. Not to mention nine tips for the DOS timid.

So, If DOS keeps asking you those hair-raising questions, like "Are you sure?"... THEN take a look at a few back issues of DOS RESOURCE GUIDE, and take the dread out of DOS!

Use our 800 number and have your credit card ready!

1-800-238-6580

Back issues of DOS RESOURCE GUIDE are $5.50 post-paid!

Or send your check or money order to:

DOS Resource Guide
P.O. Box 3162
Salisbury, MD 21802

(Please indicate what issue(s) you are purchasing.)

DON'T MISS AN ISSUE!